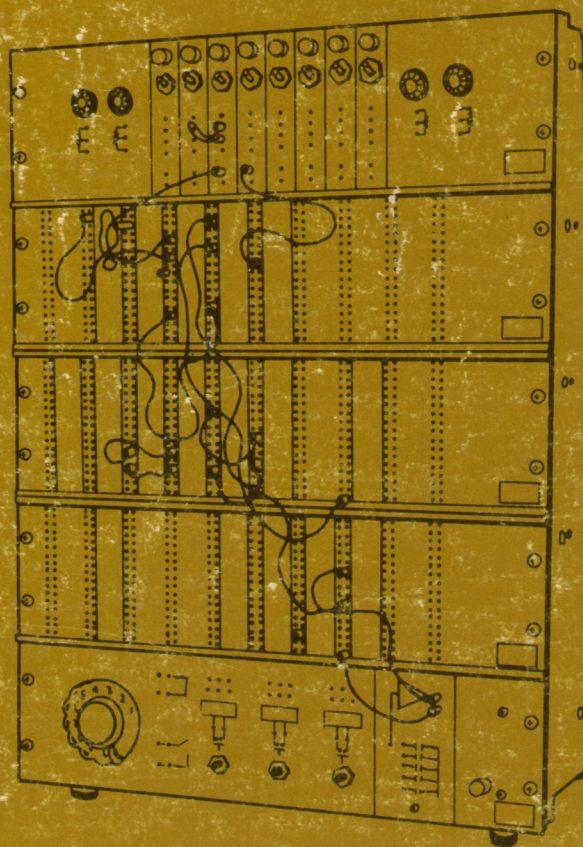


DIGITAL **LOGIC LABORATORY** **WORKBOOK**



DIGITAL EQUIPMENT CORPORATION

DIGITAL LOGIC LABORATORY WORKBOOK

Barbera W. Stephenson

Research Engineer

Copyright © 1965 by Digital Equipment Corporation

PREFACE

This workbook presents a series of laboratory experiments to accompany a lecture course in logic design, switching circuits or digital computer fundamentals. The aim of the workbook is to complement the lecture material by graphically demonstrating the principles of digital logic and by teaching some of the practical considerations not normally covered in a theoretical course.

Actually, one might consider this as an instructor's aid rather than a workbook. Each experiment has more projects and questions than could be completed in a 3 hour laboratory, so that the instructor can select those parts most appropriate in level and emphasis.

There are seven core experiments which introduce the fundamentals and build up in level. The other eight experiments, which are interspersed with the core experiments, are on a parallel level with their neighbors but cover separate topics so that they may be moved around to follow the textbook. It is hoped that the eight non-core experiments will provide enough flexibility that those experiments demonstrating basic principles can follow the corresponding lecture material and those experiments dealing with practical considerations can follow the lectures on subjects which do not lend themselves to laboratory work. Appendix E gives further notes on the experiments and relates them to leading texts.

On the other hand, to provide continuity and make a maximum contribution to the students' learning, there are areas in which the workbook is not so flexible. The core experiments build on each other and become increasingly sophisticated as the students' facility with the Logic Laboratory increases. Consequently it is suggested that these be retained in order.

Within the experiments, there are frequently a series of projects which build on a common circuit. This is done to (a) demonstrate the relationship between circuits and (b) minimize the time spent in rote wiring and thereby maximize the amount that can be learned in an experiment. These series are easily recognized and it is suggested that they be retained in order while the other questions are used to provide the desired emphasis.

The equipment used in the experiments is the Digital Equipment Corporation's Logic Laboratory, as described in Appendix E. However, some of the logic conventions used by Digital have been modified here for the benefit of the beginning students. For those who are familiar with Digital's other equipment, Appendix D outlines these modifications.

In planning this workbook, I talked with many instructors about the material that they felt should be included in the workbook and how this material could best be presented. I would particularly like to thank Charles M. Thomson, Robert Coughlin, and John Hinds of Wentworth Institute, Boston, Mass., Professor Mitchell L. Cotton of the U. S. Naval Post Graduate School, Monterey, California, and Dr. Gerard J. Stephenson, Jr., of the California Institute of Technology, Pasadena, California, for their kind assistance.

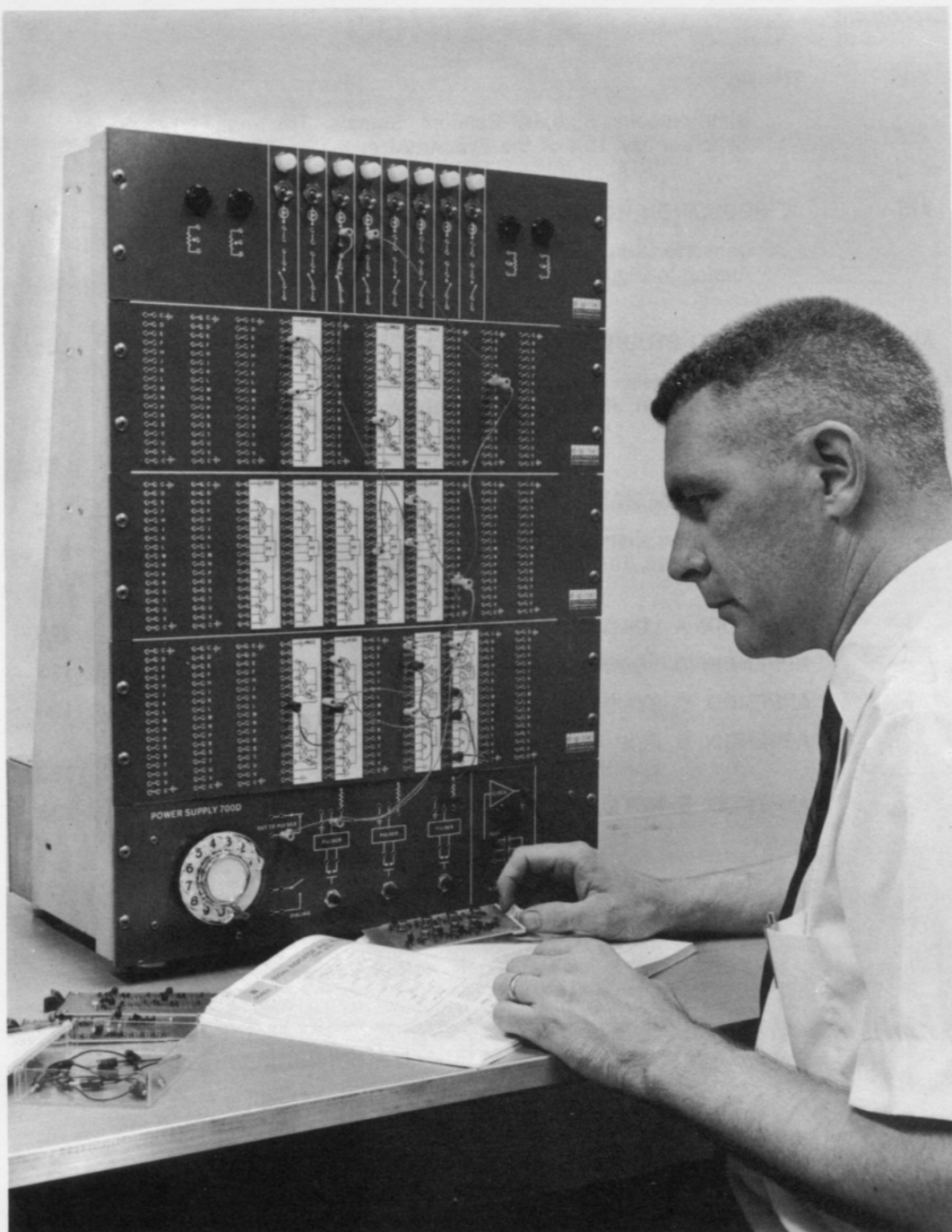
Also, many individuals at Digital Equipment Corporation offered suggestions and assistance at various stages in the writing. It is impossible to list them all here; but particular thanks go to Richard L. Best and Russell Doane for their careful reading of the drafts, and to C. Stuart Grover and Trudie Karr for their editorial assistance.

Barbera W. Stephenson
August, 1965

CONTENTS

Experiment		Page
I	BINARY NUMBERS Introduction, A Binary Counter, Use of The Clock, Negative Numbers	1
II	BINARY CODED DECIMAL Codes, An 8421 Code Counter, Excess 3 Code Counter, A Weighted, Self-Complementing Code	10
III	BASIC DIGITAL CIRCUITS — GATES The Positive NOR Gate, Symbols, The Negative NOR-Positive NAND, Gate Usage, Loading Considerations, Grounding	15
IV	FLIP-FLOPS AND DCD GATES The Flip-Flop, Diode-Capacitor-Diode Gates, Counters	22
V	FROM BOOLEAN EQUATIONS TO GATING NETWORKS Reduction Techniques	29
VI	BOOLEAN EQUATIONS AND FLIP-FLOPS Duality, Timing Considerations, Input Equations, Pulse Inputs	35
VII	ADDITION Techniques of Addition, Serial Adders, A Serial Binary Subtractor	45
VIII	PARALLEL ADDITION AND SUBTRACTION The Two-Step Parallel Adder, Positive and Negative Numbers — Two's Complement, Positive and Negative Numbers — One's Complement, Subtraction, The Single-Step Parallel Adder	50
IX	BINARY CODED DECIMAL ARITHMETIC Code Review, Arithmetic Operations with the 8421 or Excess 3 Code, Counting, Addition, Positive and Negative Numbers, Addition and Subtraction	56
X	CODE CONVERSION Code Features, Decimal Codes, Error Detecting Codes, Reflective Codes	67
XI	CONTROL The Use of Control Circuitry, Generating The Pattern, The Pulse Amplifier, The Delay (One-Shot), The Pulse Distributor Made With Delays	75

Experiment		Page
XII	TIMING Simultaneous Signals, Random Signals, The Synchronizer, Use of the Synchronizer	85
XIII	INTRODUCTION TO ANALOG-DIGITAL CONVERSION Converter Uses, Digital-to-Analog Conversion, Analog-to-Digital Conversion, The Continuous Converter	98
XIV	ADVANCED STUDIES IN ANALOG-DIGITAL CONVERSION The Successive Approximation Counter, Comparison of Analog-to-Digital Converters, Accuracy	105
XV	COMPUTER DESIGN Computer Elements, General versus Special Purpose Computers, The Algorithm, Hardware	114
	APPENDIX A LOADING RULES 125	125
	APPENDIX B ADDING WAVEFORMS 127	127
	APPENDIX C SYMBOLS 129	129
	APPENDIX D FLIP CHIP MODULES IN THE LOGIC LABORATORY 130	130
	APPENDIX E NOTES TO THE INSTRUCTOR 133	133



The Digital Logic Laboratory

EXPERIMENT I

BINARY NUMBERS

PART 1 INTRODUCTION

A digital computer is an assemblage of extremely simple circuits. Consider the familiar elements in the logic laboratory—the toggle switches and push buttons. Examine these elements in detail. Compare them.

Each has only two states. The switch may be up or down. The button may be depressed or released. Digital circuits also have two states, a negative voltage level and a positive level. In the logic laboratory the negative level is -3 volts; the more positive level is ground.

Note the differences. The button makes contact only when depressed. When released, it always returns to its original position. The switch, by contrast, always remains where last positioned. It remembers. In this same sense, digital circuits are divided into two classes, those which remember and those which follow.

Employing only two state devices, with and without memory, one can construct a complex computer system, which is able to control the motion of a missile, schedule production in a diversified factory, or compose symphonic music.

To design a computer from these basic concepts is similar to proving a complex mathematical theorem from a simple set of axioms. From bare axioms, one proves a modest theorem; from this, a more intricate theorem; and finally, a sophisticated system of theorems.

In designing computer logic to solve a specific problem, just as in developing a mathematical proof, many approaches may be taken. Some may be superior because they are more efficient; others may be easier to use. But, frequently, many different techniques are equally acceptable. The logic laboratory provides a means, not only to try accepted techniques, but also to try your own designs.

PART 2 A BINARY COUNTER

How can two state devices represent the immense quantities required to calculate the orbit of a satellite? To balance the books of a multimillion dollar bank? Or to tabulate votes and predict an election outcome?

The binary number system, as described in your text, is a preferred solution because it is efficient, each number has a unique representation, and simple rules define the arithmetic operations.

In this experiment, you will study the binary number system by constructing a counter, using the Flip-Flop Type R201. The flip-flop is the circuit equivalent of the toggle switch. It remembers.

The flip-flop circuit is shown symbolically in Figure 1. The two outputs are always in opposite states. That is, if one output is at -3 volts, the other is at ground and vice versa. To see this, connect the two flip-flop outputs to indicator lights. One will be on while the other is off.

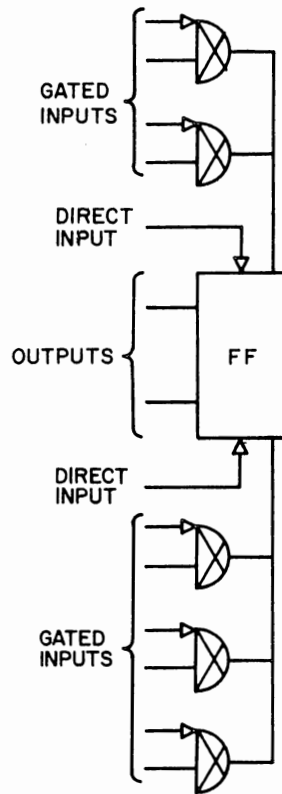
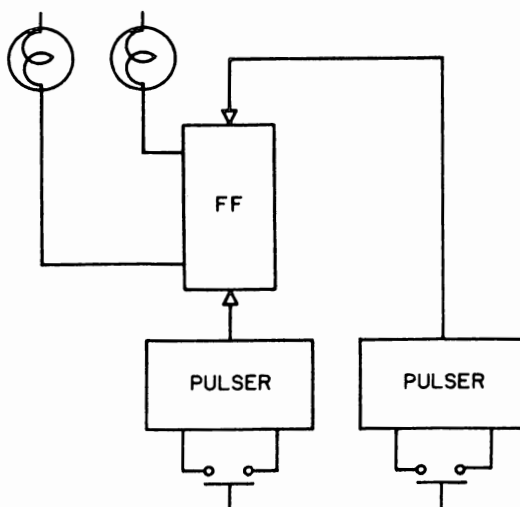


Figure 1 Symbolic Representation of a Flip-Flop

To change the state of the flip-flop, connect a push-button pulser to each of the direct inputs as shown in Figure 2. Notice that the flip-flop always remembers which button was depressed last.



NOTE on Pulser Connection: When connecting a pulser output to a flip-flop or other module, the ground pin next to the pulser output should be connected to the ground on the module being driven.

Figure 2 Changing the State of a Flip-Flop

Since the upper and lower halves of the flip-flop module are indistinguishable (except for one extra gate on the lower side), it is convenient to name them. Because a counter works with numbers, the labels ONE and ZERO seem appropriate. If the upper half is labeled ONE, the upper output, direct input, and gated inputs may all be designated as ONE terminals. Similarly, all inputs and outputs on the lower half will be designated as ZERO terminals. This is shown in Figure 3 and may be marked in the corner of the symbol for convenience.

Because of the complementary nature of the flip-flop outputs, the two indicator lights are redundant. If the indicator is removed from the ZERO output terminal, the state of both outputs is still specified by the ONE indicator. When the ONE indicator is lit (and the ONE output is at ground), the flip-flop will be said to be in the ONE state. When the light is off (and the ONE output is at -3 volts), the state of the flip-flop is ZERO.

To construct a counter of 0 to 63, label six flip-flops with ZEROs and ONEs as shown in Figure 3. Remove all previous connections and tie each ONE output terminal to an indicator.

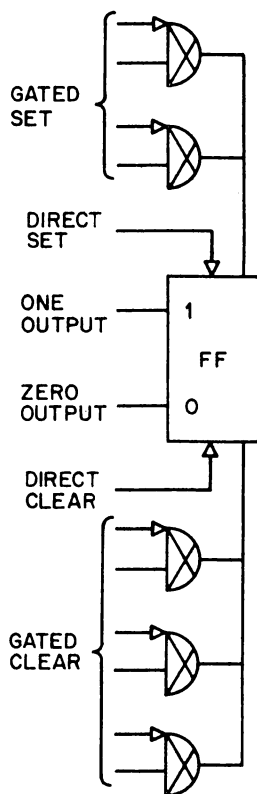


Figure 3 Designations

Since the flip-flop comes up in a random state when the power is turned on, some indicators will be lit, while others will not. To provide a clear signal, jumper all the direct ZERO input terminals to the push-button pulser. The connections should appear as in Figure 4. When the push button is activated, all the lights should go off. If this does not happen, check your wiring.

To distinguish among the flip-flops, label them from right to left according to the weights used in the binary number system and the labels in Figure 4.

Jumper the gates of the right-hand flip-flop to the dial through a pulser as shown in Figure 5. Use the dial terminals marked "out to pulser."

The two gate terminals with arrowheads connect to the dial pulser. The two without arrowheads are grounded. Test the wiring by dialing 1. This should change the state of flip-flop 1.

Connect the next flip-flop as shown in Figure 6.

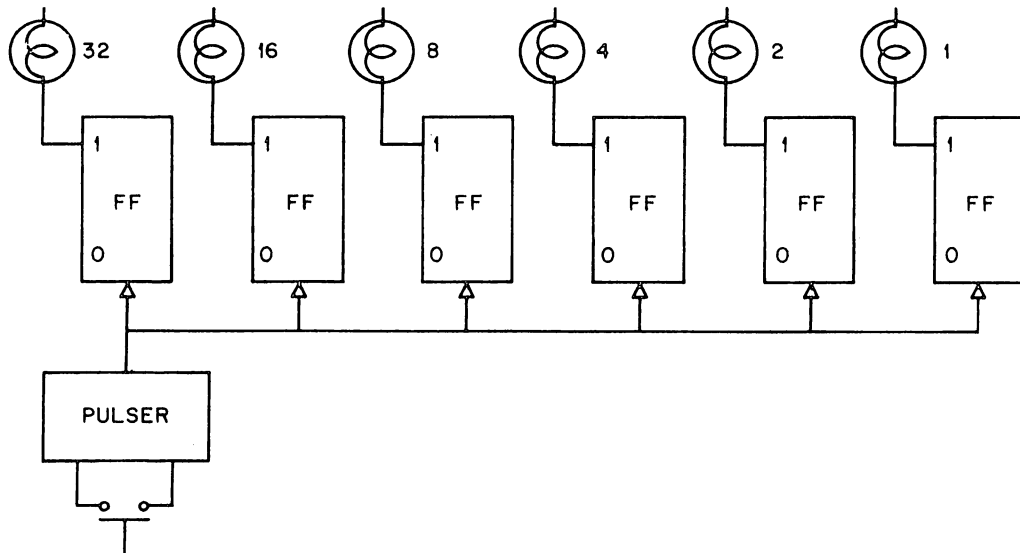


Figure 4 Clearing the Counter

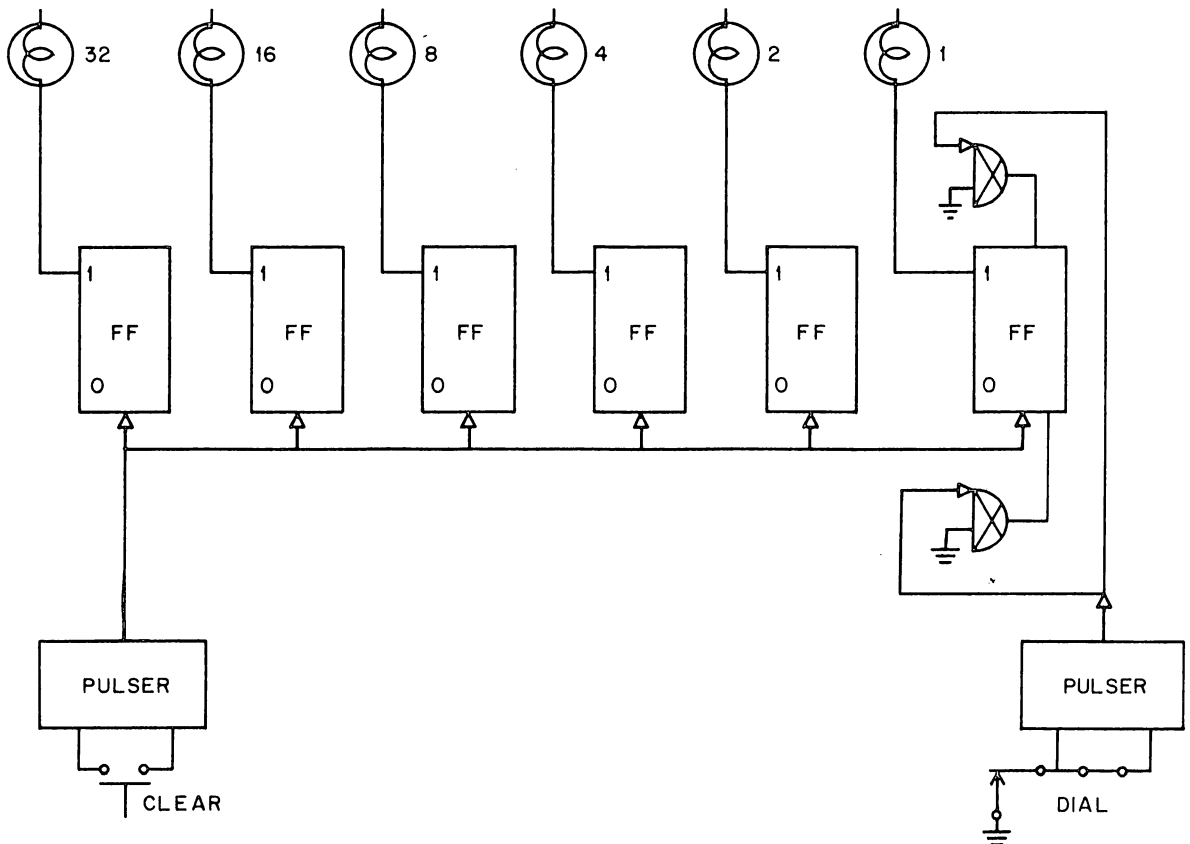
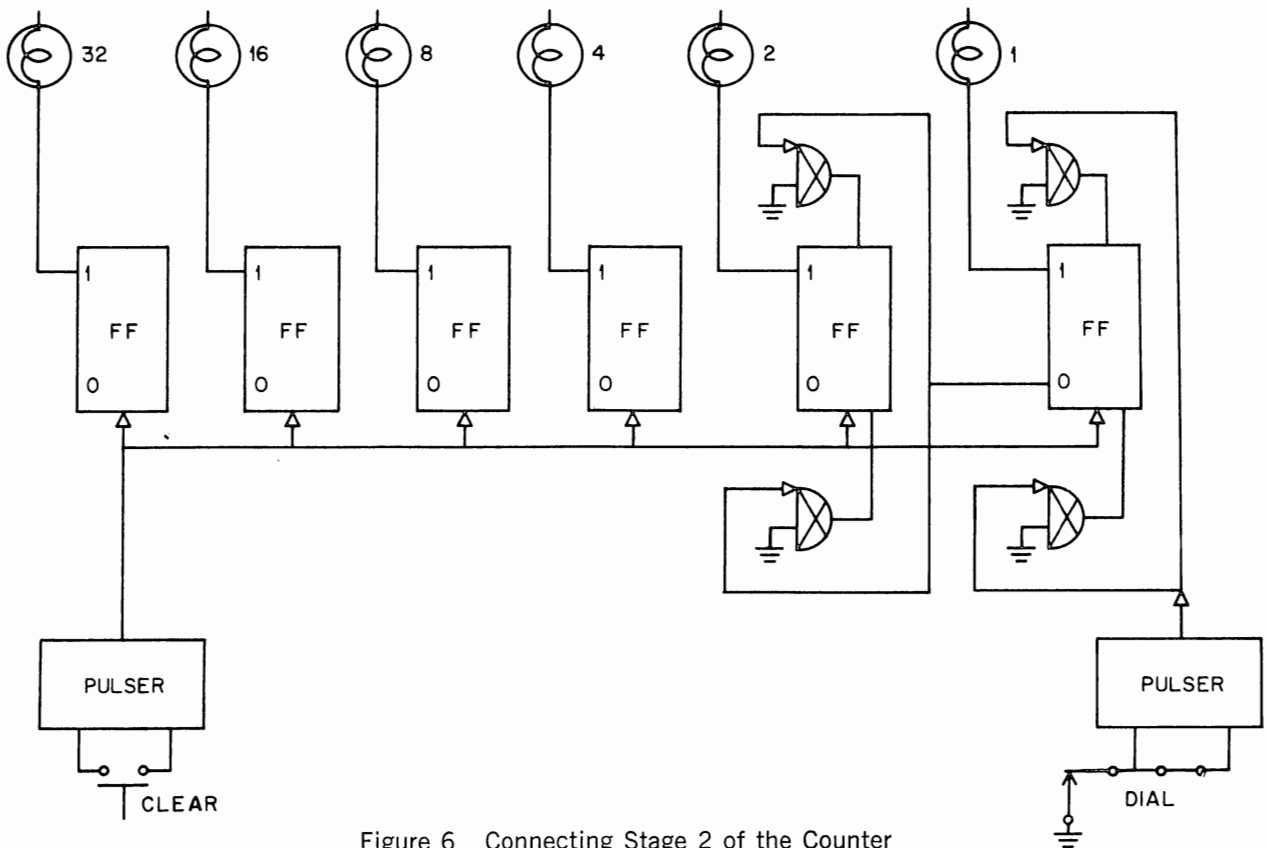


Figure 5 Connecting Dial to Flip-Flop 1



Test your wiring by clearing the counter, which should turn off all the indicators, then dialing 2, which should light the second indicator lamp only. As you turn the dial, the first indicator light will blink, showing that you passed through the number 1 in going to the number 2.

Figure 7 shows the complete counter circuit. Wire in stage number 4 and test it by clearing and dialing 4. Continue this procedure with stages 8, 16, and 32. Test them by dialing 8, by dialing 8 twice, and by dialing 8 four times. When the wiring is correct, the counter is complete.

The counter accepts inputs from the dial, and the indicators display the accumulated total. Since the dial is coded in decimal while the counter and indicators are in binary, the circuit also performs decimal to binary conversion.

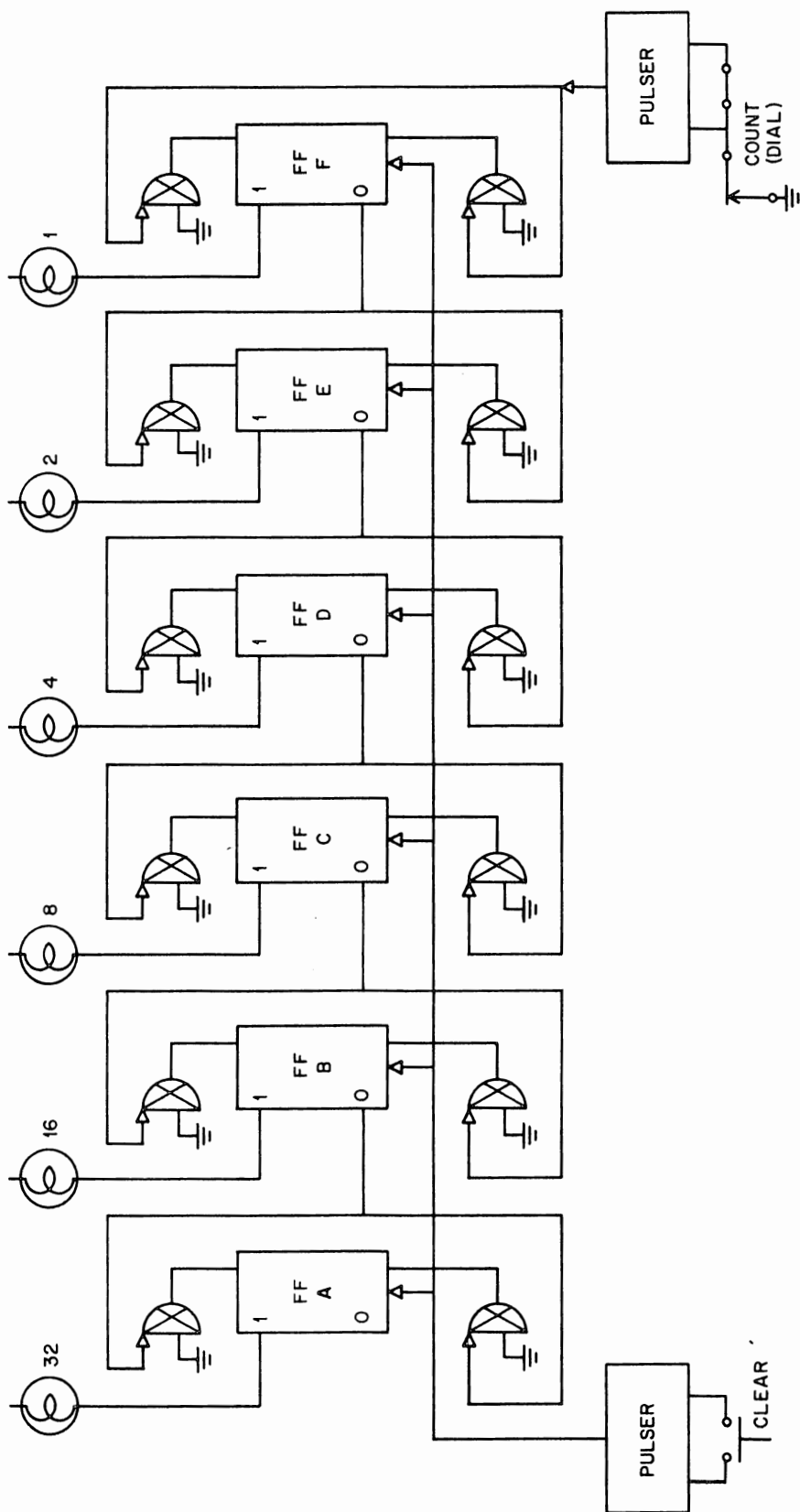


Figure 7 Binary Counter

1. What are the binary equivalents of the following numbers?

Decimal	Binary
5	
10	
12	
17	
19	

Decimal	Binary
23	
37	
42	
59	
61	

2. What happens when you dial 64? 65? 66? What causes this?

PART 3 USE OF THE CLOCK

By replacing the dial input with a high frequency pulse source, you can observe the continuous behavior of the counter. The pulse source is the clock circuit, located near the push buttons. Remove the dial pulser from the counter input and replace it with the clock. Run a wire from the clock ground to the flip-flop ground as you did with the pulser.

3. Observe the clock output on the oscilloscope and sketch the waveform.

NOTE: On oscilloscope settings. Since the circuits are extremely fast, a low capacitance x10 probe should be used. All standard logic kit signals are between ground and -3 volts, hence the gain setting with the x10 probe should be 0.1 to 0.25 volts/inch or 0.05 to 0.1 volts/cm. The frequency of the clock will be between 200 kc and 2 mc. To see several periods, the horizontal sweep setting should be 50 kc to 500 kc, or 0.5 to 5 $\mu\text{sec/inch}$ or 0.25 $\mu\text{sec/cm}$. For good resolution of a single pulse, the horizontal sweep should be approximately 5 mc or 50 $\mu\text{sec/inch}$ or 25 $\mu\text{sec/cm}$. The synchronization selector may be set on + INTERNAL.

4. Adjust the clock pulses to a frequency of 2 mc, then observe the ONE output of flip-flop 1. Sketch the waveform. What is the frequency? Why is the waveform symmetrical?
5. Observe the ONE outputs of flip-flops 2, 4, 8, 16, and 32. What is the frequency of each?
6. If an additional stage were added to the counter, what would the frequency of its output be?
7. A counter is often called a scaler. Explain why this name is applicable.

PART 4 NEGATIVE NUMBERS

Move the indicator connections from the ONE sides of the flip-flops to the ZERO sides so that the clear button lights all the indicators. In unsigned binary, this number is 63. In 1's complement, it is interpreted as -0 . In 2's complement, it is interpreted as -1 .

Dial 5. The lights read 111010. In unsigned binary, this is 58. In 1's complement, it reads -5 . In 2's complement, it translates as -6 . Simply by moving the indicator wires, the counter has been changed into a down counter.

8. Dial the following numbers and record the outputs. What is the decimal equivalent of the results as interpreted in each of the three conventions?

Number Dialed	Output	Equivalent Decimal if the convention is:		
		Unsigned Binary	1's Complement	2's Complement
Clear	111111	64	—0	—1
1				
4				
7				
8				
10				
14				
32				

PART 5 SPECIAL PROBLEMS

9. What do the indicators, the toggle switches, the push buttons, and the computer elements all have in common?

10. What is the difference between the push button and the toggle switch? Which does the flip-flop resemble?

11. The binary number system is not the only method of coding 2-state devices to represent decimal numbers. A single 2-state element, such as an indicator, can be coded in two ways to represent the numbers 0 and 1, that is:

State of Indicator	Codes	
off	0	1
on	1	0

(a) With two indicators, there are four possible states. How many different ways can these be coded to represent the numbers 0, 1, 2, and 3? Complete the table below:

States		Codes																			
off	off	0	0	0	0	0	0	1	1	1	1										
off	on	1	1	2	2	3	3	0	0	2											
on	off	2	3	1	3	1	2	2	3												
on	on	3	2	3	1	2	1	3													

There are 24 ways in which these two devices can be coded. In statistics this is expressed by saying that the combinations and permutations of four things, taken four at a time, is $4! = 24$. You can visualize this as follows: The off-off state can correspond to any of the four numbers. For each of these four, there are three remaining numbers which can be assigned to the off-on state. This gives $4 \times 3 = 12$ possibilities. For each of the twelve possibilities, there are two remaining numbers which can be assigned to the on-off state. This yields $12 \times 2 = 24$ possibilities. For each of the 24 possibilities, there is one remaining number which can be assigned to the on-on state. The total number of codes is, then, $4 \times 3 \times 2 \times 1 = 24$.

For three indicators, there would be eight states. The total number of possible codes would be $8! = 8 \times 7 \times 6 \times 5 \times 4 \times 3 \times 2 \times 1$.

(b) How many states are there for four indicators? How many possible codes?

(c) With this many choices, it would not be practical to try all. It is best to select a code convention where the representation for each number is determined by a fixed set of rules. List the code conventions you have studied so far.

(d) List the rules for determining the code for a particular number in the binary system.

(e) What are the advantages of the binary number system?

12. Two state devices occur frequently. Consider the problem of tossing a coin. It can come up either heads or tails. If the coin is tossed twice, what is the probability that one toss will come up heads and one will come up tails? Bear in mind that the four possible outcomes are:

tails-tails
tails-heads
heads-tails
heads-heads.

(a) If the coin is tossed three times, how often will the sequence "heads-heads-tails" occur in that order?

(b) How many times will heads come up twice and tails come up once?

13. Computers are frequently used for sorting and alphabetizing. A manual card sorter can be constructed with index cards, a hole-punch, scissors and a pencil. Number the cards, and punch a row of holes near one edge. Then design a system for coding the cards with the scissors. Without looking at the numbers written on the cards, how could you use the pencil or a toothpick, to select card number 7?

(a) Card number 5?

(b) All even cards?

(c) Could you develop a similar coding system for letters of the alphabet?

(d) What practical uses might such a system have?

EXPERIMENT II

BINARY CODED DECIMAL

PART 1 CODES

In a binary coded decimal system, every decimal digit is coded separately. For example, the number 514 is represented by a code for 5, followed by a code for 1, followed by a code for 4. Because this type of coding system is particularly easy to convert to decimal, it is used in computers which must frequently read and write decimal numbers.

Figure 1 shows two examples of binary coded decimal (BCD) which will be studied in this experiment.

Decimal	8 4 2 1	Excess 3
0	0 0 0 0	0 0 1 1
1	0 0 0 1	0 1 0 0
2	0 0 1 0	0 1 0 1
3	0 0 1 1	0 1 1 0
4	0 1 0 0	0 1 1 1
5	0 1 0 1	1 0 0 0
6	0 1 1 0	1 0 0 1
7	0 1 1 1	1 0 1 0
8	1 0 0 0	1 0 1 1
9	1 0 0 1	1 1 0 0

Figure 1 8421 and Excess 3 Codes

The 8 4 2 1 code uses the same representations as the binary number system. The name of the code is derived from the weights (or positional values) of the bits. That is, the decimal equivalent of a number can be determined by adding the weights of those positions in which a ONE occurs. The operations of counting, adding, and multiplying are similar to binary except that the states representing the numbers 10 through 15 are not used. Subtraction, however, is more difficult.

In the Excess 3 (XS3) code, a decimal number D is represented by the binary equivalent of the number D plus 3. The Excess 3 code is not weighted; but, since it follows the same number sequence as binary, it is useful in arithmetic operations. It is more useful for subtraction than the 8421 code since the 9's complement of any digit can be obtained simply by complementing each bit.

PART 2 AN 8421 CODE COUNTER

The complete diagram for the 8421 code counter is shown in Figure 1. Label four flip-flops according to the weights, 8421. Jumper the ONE output terminal of each flip-flop to an indicator light. To provide a clear signal, connect the push-button pulser to all the ZERO input terminals. Connect a ground wire. Check that the push button clears all of the flip-flops.

Complete all of the wiring, then test it according to the test table below:

Test Table

Number Dialed	Indicators
clear	0 0 0 0
1	0 0 0 1
2	0 0 1 0
4	0 1 0 0
8	1 0 0 0

1. What is the 8421 code representation for the number 5? For 7? For 9?
2. What happens when you dial 10, 11, 12, 13, 14, or 15?
3. Is the state 1111 ever used?
4. Replace the dial input with a clock signal. Referring to the notes on "use of the clock" in Experiment I, adjust the clock pulses to a frequency of 2 mc. Observe the ONE output of flip-flop 1 and sketch the waveform. What is the frequency? Why is the waveform symmetrical?

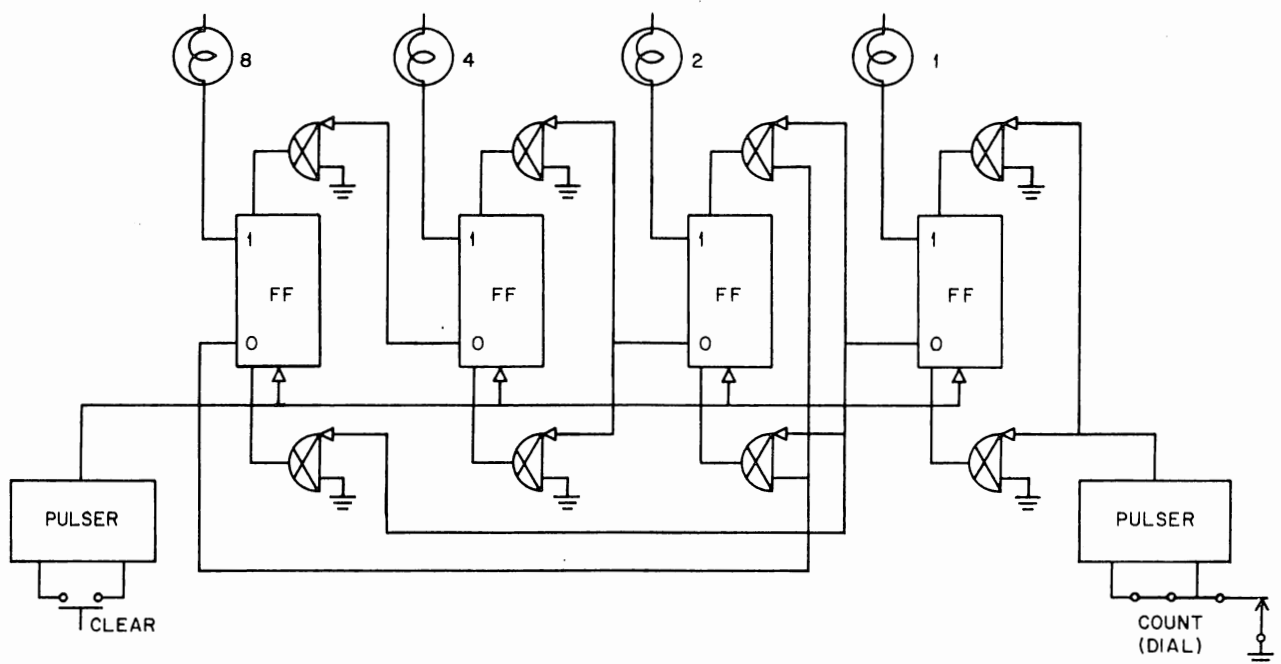


Figure 2 8421 Counter

5. Observe and sketch the ONE output of flip-flop 8. What is the frequency? Why is the waveform not symmetrical?
6. What result would you anticipate if you moved all the indicator connections from the ONE to the ZERO side of the flip-flop as in Experiment I? Could the result be interpreted as a down-counter in 8421 code? Why or why not?
7. After answering question 6, try moving the indicator connections. Use the dial as the input and record the results.

PART 3 EXCESS 3 CODE COUNTER

Figure 3 shows an up counter using the Excess 3 code. Construct the circuit and test the wiring using the following conditions:

Test Table

Dial	Indicators
clear	0 0 1 1
1	0 1 0 0
2	0 1 0 1
3	0 1 1 0
5	1 0 0 0

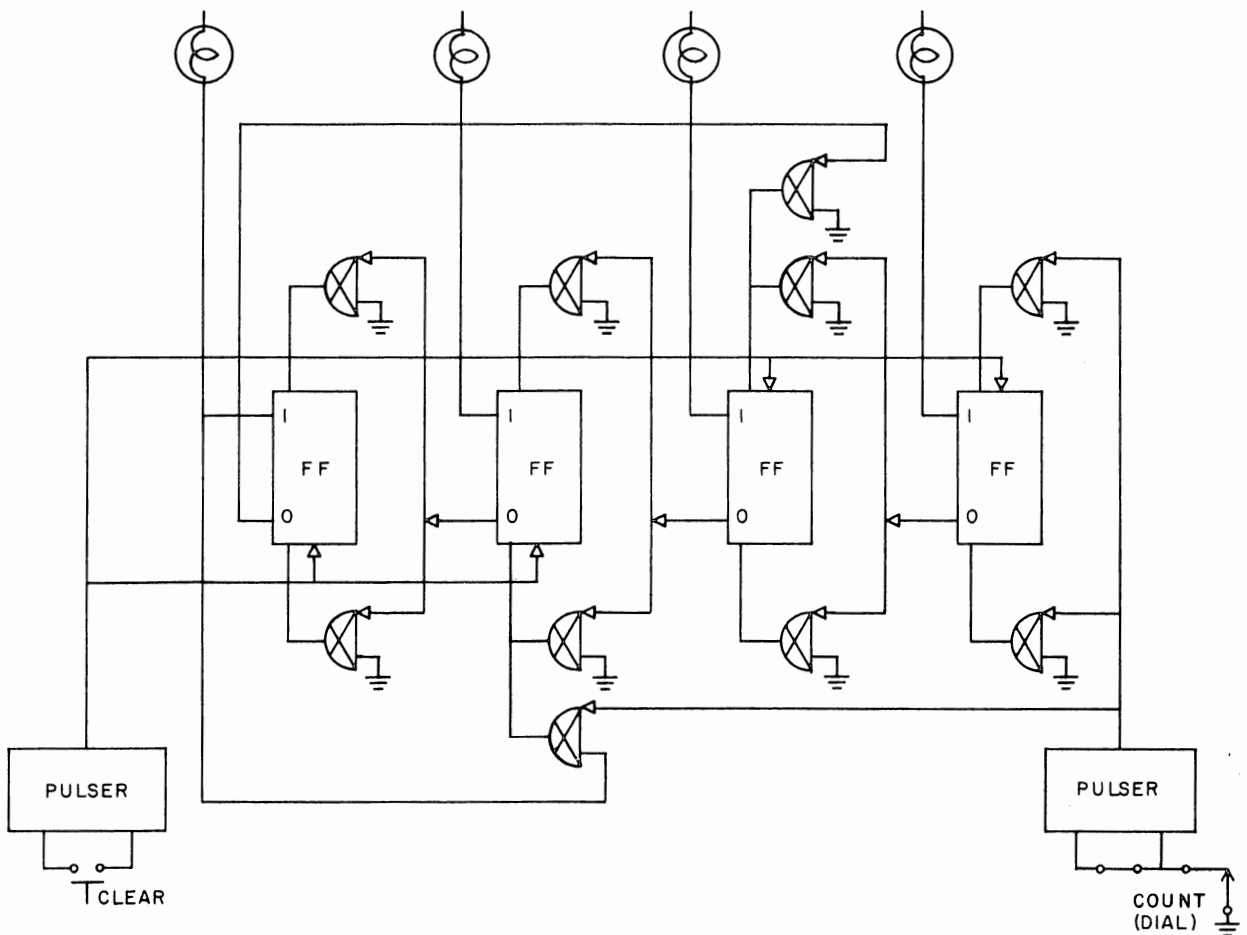


Figure 3 Excess 3 Code Counter

8. Why does the clear signal light two of the indicators?
9. How are the numbers 0 through 9 represented in this code? Check the codes by clearing, then dialing the selected number.
10. Dial 9, then dial 1. Explain the action of the circuit when this tenth pulse is sent into the counter.
 - (a) What happens when you dial a series of numbers where the sum is greater than ten?

11. If you moved the indicator wires from the ONE terminal of the flip-flops to the 0 terminal, what would happen? Test the circuit and find out.

PART 4 A WEIGHTED, SELF-COMPLEMENTING CODE

Figure 4 shows an up counter for another common code. Construct the counter and test it according to the test table below:

Test Table

Dial	Indicators
clear	0 0 0 0
1	0 0 0 1
2	0 0 1 0
4	0 1 0 0
6	1 1 0 0

12. What are the code representations of the ten decimal digits? Does this code group follow the same sequence as the binary number system?

13. It is possible to assign weights to each of the binary bits. What are these weights?

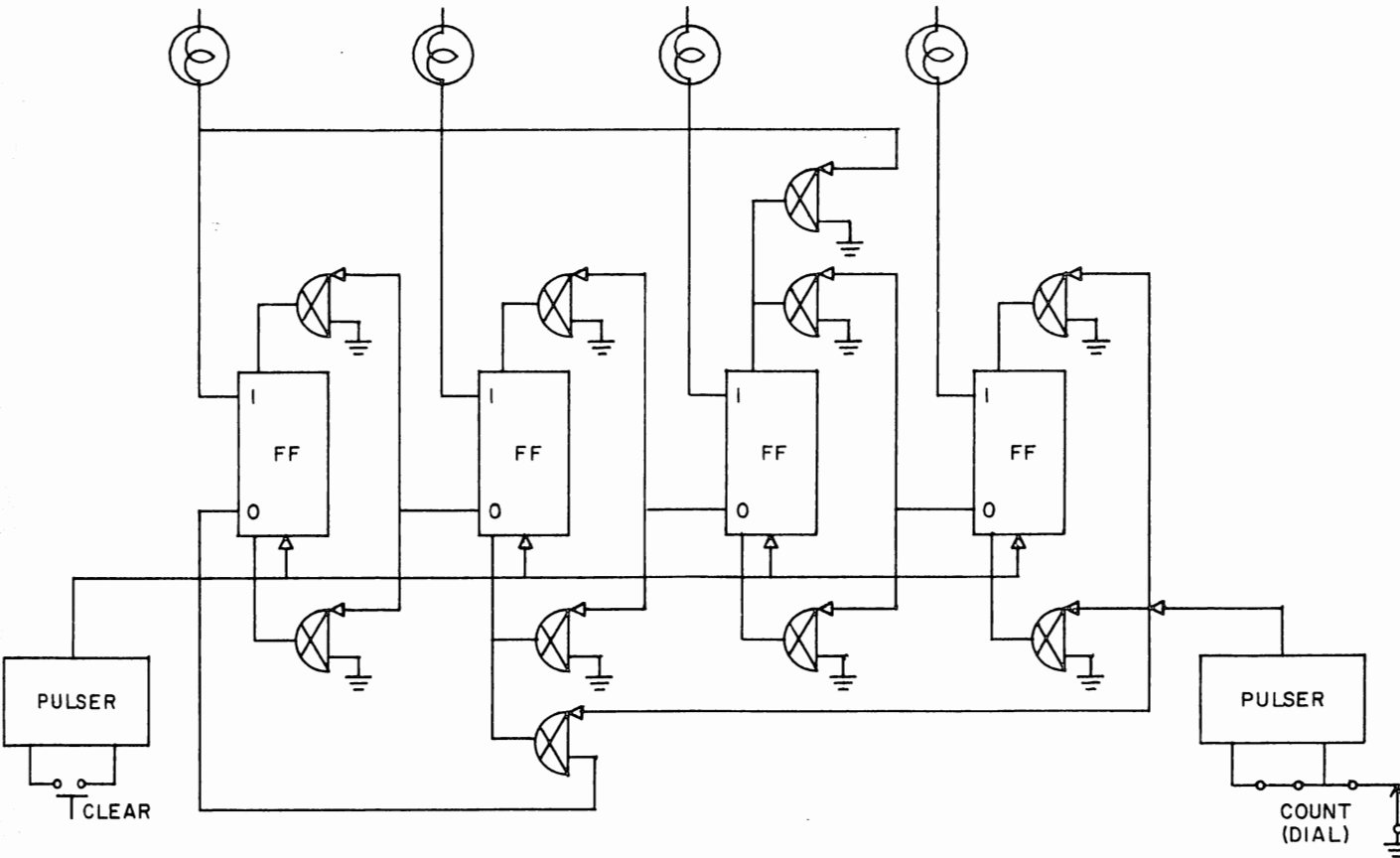


Figure 4 Counter for a Weighted Self-Complementing Code

14. If you change the indicator wires from the ONE terminal to the ZERO terminal on the flip-flops, will you obtain a down counter which operates on the same code group? Why or why not?
15. Test your answer to question 14 and record the results.
16. How many different codes are possible when four binary bits are used to represent the ten decimal digits? Hint — There are 16 possible states which can be used to represent the first digit. For each of these 16, there are 15 states which can be used to represent the second digit.
17. Make a code where the binary weights are 5311. How many different code groups can you make this way?

EXPERIMENT III

BASIC DIGITAL CIRCUITS—GATES

PART 1 THE POSITIVE NOR GATE

This basic circuit element in the Logic Laboratory is the simple diode gate shown in Figure 1. Most of the other circuits in the Logic Laboratory can be constructed with minor variations on this circuit.

Logically, the diode gate is equivalent to the push button. It follows. However, the diode gate follows in such a manner that it performs many useful functions. It combines, amplifies, inverts, and standardizes the signals which represent various logical functions.

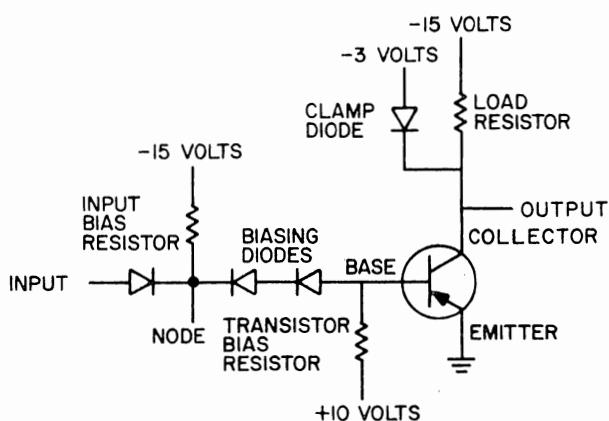


Figure 1 Single-Input Diode Gate

From the schematic of Figure 1, you can learn the operation of the diode gate. When the input is negative, the node is also negative and current flows from the transistor emitter through the biasing diodes and the input bias resistor to -15 volts. As a result, the PNP transistor is turned on forming a short circuit between the collector and the emitter. Thus, when the input is negative, the output is at ground. Since the output is from a saturated transistor, it has a low output impedance and good driving power.

When the input is at ground, the biasing diodes and the transistor bias resistor hold the transistor base more positive than the emitter, and the transistor is turned off. The transistor then acts as an open circuit and the output voltage is determined by the load resistor and clamp diode. These serve as a voltage source, holding the output at -3 volts.

The single-input diode gate therefore serves three functions. It inverts the input signal. It standardizes the output voltage to ground or -3 volts. Since the output current available from the transistor is much greater than the required input current, the diode gate also amplifies.

A fourth function, gating, is obtained by adding more diode inputs to the node as shown in Figure 2.

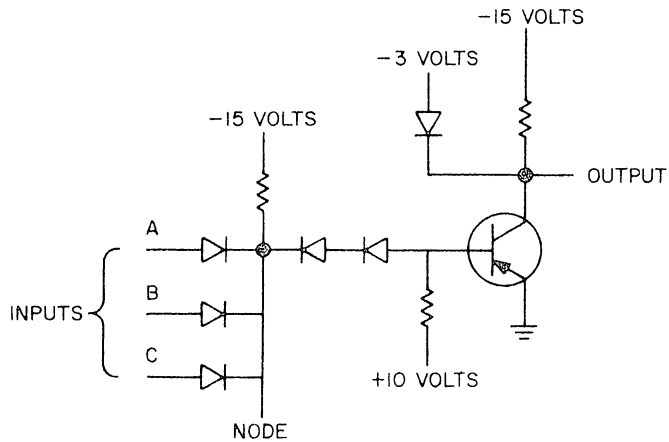


Figure 2 Multiple-Input Diode Gate

The node will be at approximately the same voltage as the most positive input. Thus, when any input terminal is grounded, the node is also at ground and the circuit output is at -3 volts. If all of the inputs are negative, the node terminal will be negative and the circuit output will be at ground.

This circuit is frequently called a positive NOR, which stands for positive-input, negated OR. The name derives from the fact that, if A OR B OR C is positive, the output will be at the opposite voltage level, i.e., -3 volts. In the same manner, this circuit is also called a negative NAND, for negative-input, negated AND. If A AND B AND C are all negative, the output will be at ground.

PART 2 SYMBOLS

Although the basic diode gate can be used to construct very complex logical functions, a diagram that showed all the circuit components would be tedious to draw and difficult to read. For this reason, the drawings employ a shorthand notation which represents one or more components as a single functional unit. Referring to Figure 3, the gating is shown as a semicircle with a single diode inside. The circle at the output represents the inversion function.

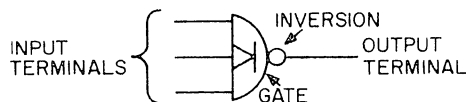


Figure 3 Positive NOR Symbol

PART 3 THE NEGATIVE NOR- POSITIVE NAND

It is also useful to have a circuit which performs the same function on the opposite voltage levels; that is, a negative NOR, positive NAND. In this circuit, if input A OR B OR C is negative, the output will be at ground; if A AND B AND C are positive, the output will be at -3 volts.

The Logic Laboratory circuit, which performs this function, is quite complex; however, one can picture this gate by considering the circuit of Figure 2 with the input diodes reversed and the input bias resistor returned to a positive, instead of a negative, voltage. From this simple picture comes the logic symbol for the negative NOR. It is the same as for the positive NOR, but the input diode is reversed. This symbol is shown in Figure 4.

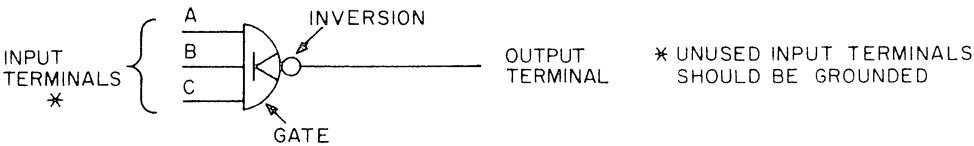


Figure 4 Negative NOR Symbol

PART 4 GATE USAGE

1. To study the operation of the diode gate, form a square wave signal from a clock and a flip-flop and drive a diode gate as shown in Figure 5. Sketch the waveforms at the diode gate input and output.

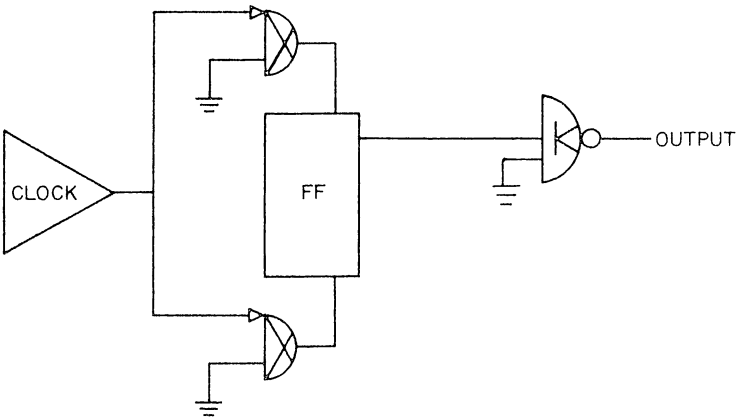


Figure 5 Diode Gate Driven by Square Wave

2. To make pulses to appear at the output only when a control switch is closed, connect a toggle switch as shown in Figure 6. What happens when the toggle switch is up? When it is down?

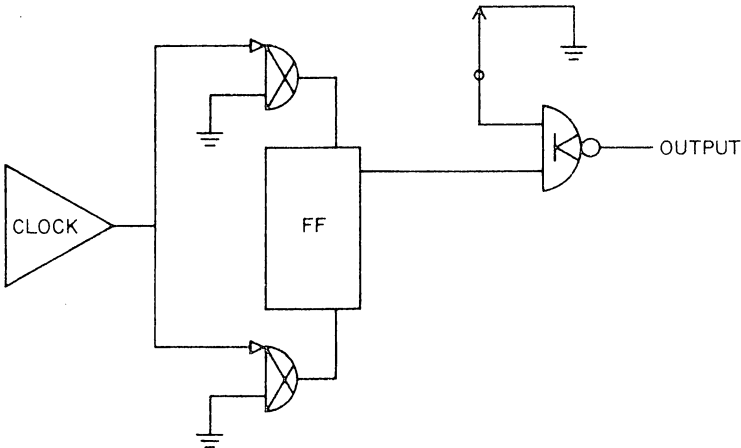


Figure 6 Gated Square Wave

3. Frequently it is necessary to send signals to more than one place. Wire in a second gate as shown in Figure 7. What toggle switch settings will produce square waves on the following lines? (Closed = 1)

Switches		Outputs	
Sw1	Sw2	L1	L2
0	0		
0	1		
1	0		
1	1		

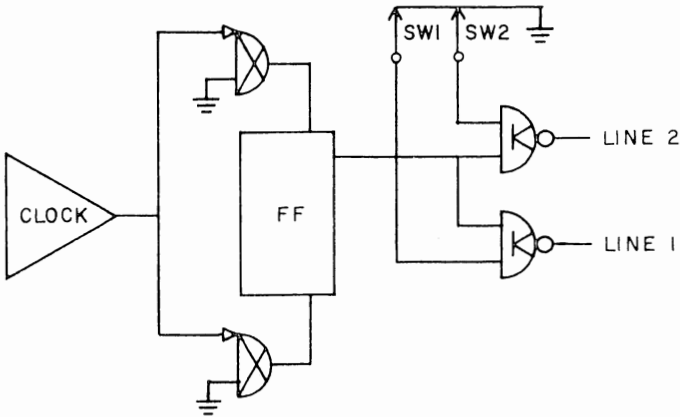


Figure 7 Pulse Distributor

The circuit of Figure 8 forms a decimal to binary converter. If the switches are labelled 1-7, and only one switch is selected, then the output will correspond to the binary representation of that number. For example, if switch 1, 3, 5, or 7 is closed, the output of gate A (a positive NOR) will be a negative voltage. This is applied to a negative NAND gate along with the square wave. When the square wave is negative, both inputs will be negative, and the output will go to ground. When the square wave is at ground, the AND condition is no longer met, and the output will go negative. Thus the output will be a square wave which is 180 degrees out of phase with the flip-flop.

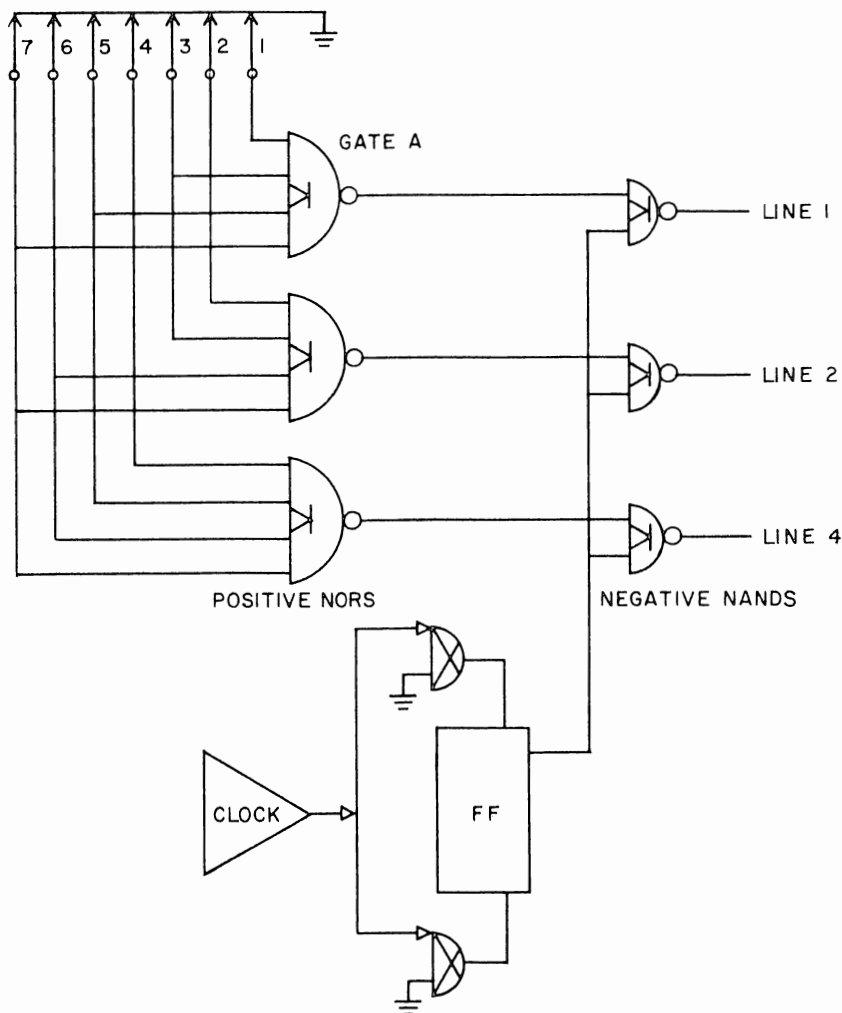


Figure 8 Decimal to Binary Converter

4. What are the conditions for generating an output on Line 2? On line 4?
5. Construct the circuit and observe the operation of the circuit as a decimal to binary converter and as a general decoding network. On which lines would the output appear if the following combinations of switches are closed?

Switches	Outputs
1 and 5	
1 and 6	
2 and 4	
5 and 6	
4, 5, and 6	
3 and 7	

- (a) Extend this circuit to include conversion of the number 8.

6. Design and test a circuit which will gate through the square wave only if both switches 1 and 2 are closed. (Hint: use a positive NAND and a negative NAND.)
7. Design and test a circuit that will gate through a square wave according to the following:
- (a) Line 1, if Sw1 or Sw2 is closed.
Line 2, if Sw2 is closed.
 - (b) Line 1, if any odd numbered switch is closed.
Line 2, if Sw2 or Sw7 is closed.
Line 3, if Sw2 and Sw7 are both closed.
 - (c) Line 1, if push button is depressed and either Sw1 or Sw2 is closed.

PART 5 LOADING CONSIDERATIONS

The output of the diode gate circuit is either held at ground by the saturated transistor or is held at -3 volts by the clamp diode and load resistor. However, if too much current is drawn from the output, either the transistor or the clamp diode will come out of saturation and the voltage levels will deteriorate. In this case, the signals will no longer be proper for driving other units. Thus, it is important that the circuits never be overloaded.

The input current required for correct circuit operation is called the “load” presented by that circuit. The current which an output can supply is called the driving ability of the circuit. When two output terminals are connected, one output may also present a load to the other. A table of the loads and load driving abilities of some of the modules is shown below.

Terminal	Load	Driving Ability
Inputs		
Diode Gate	1 ma	
Indicator	1 ma	
Special Load (above pulser)	10 ma	
Outputs		
Diode Gate	2 ma	18 ma
Pulser	2 ma	18 ma
Flip-Flop (upper side)	7 ma	13 ma
Flip-Flop (lower side)	9 ma	11 ma
Clock	3 ma	70 ma

The load driving ability listed in the table always refers to external loads. For example, the clock module is capable of driving 70 ma of external load in addition to its own internal load of 3 ma.

At any given time, the actual load driving ability of the majority of circuits will greatly exceed the rating. However, the engineer should always design his circuits to be within the rated loading for the following reasons:

1. Circuits should be interchangeable in the design. The loading restrictions provide a minimum standard which all circuits meet. If the designer exceeded this minimum, he would have to test his circuits to select the best

ones—a rather tedious process. Similarly, if a circuit were damaged, the service personnel would have to perform special tests.

2. Driving ability will change with time and with the conditions under which the circuit is used; therefore, a margin of tolerance must be allowed. Age, amount of use, amount of power dissipated, temperature, humidity, and changes in the power supply voltages will all change the circuit characteristics.

8. Calculate the loading in each of the following cases:

- (a) On the flip-flop in Figure 5.
- (b) On switch 1 in Figure 7.
- (c) On the flip-flop in Figure 7.
- (d) On switch 1 in Figure 8.
- (e) On switch 7 in Figure 8.
- (f) On the flip-flop in Figure 8.

PART 6 GROUNDING

Like loading considerations, grounding is not a part of the logical operation of a circuit, but it is an important consideration in the actual construction of a circuit. In the Logic Laboratory all circuits are grounded through the power supply. However, for signals requiring a sharp transition this is a long path. A separate ground return should be run between the two modules if the distance is long. The rule is:

Use a separate ground return with a

pulser output
clock output, or
inputs with a \rightarrow symbol

if the wire must travel

between two mounting panels or
across a mounting panel.

EXPERIMENT IV

FLIP-FLOPS AND DCD GATES

PART 1 THE FLIP-FLOP

The flip-flop is the logic circuit which stores information. It can be constructed very simply by cross coupling two diode gates, as shown in Figure 1. If one of the outputs is at ground, it holds the other output negative. This in turn holds the first output at ground.

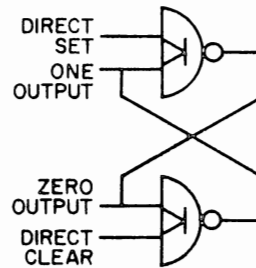


Figure 1 The Flip-Flop

Thus, the flip-flop holds itself in one of two stable states, depending on which output is at ground.

The state of a flip-flop may be changed by grounding one of the output terminals as shown in Figure 2. It may also be changed by grounding one of the other diode gate inputs, called direct inputs. This is shown in Figure 3.

1. Construct a flip-flop as shown in Figure 4. Connect a push button to the direct clear terminal and the dial to the direct set terminal. What happens when the button is pushed? When the dial is turned?

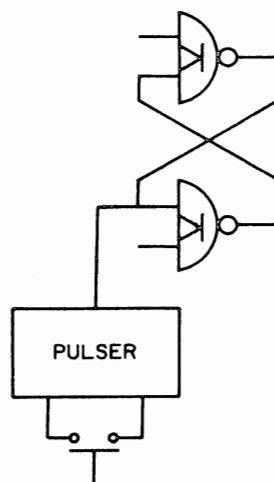


Figure 2
Clearing a Flip-Flop
Through the Output

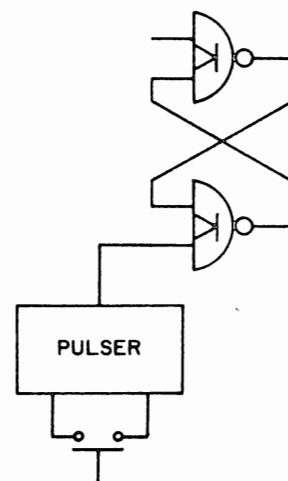


Figure 3
Clearing a Flip-Flop
Through Direct Inputs

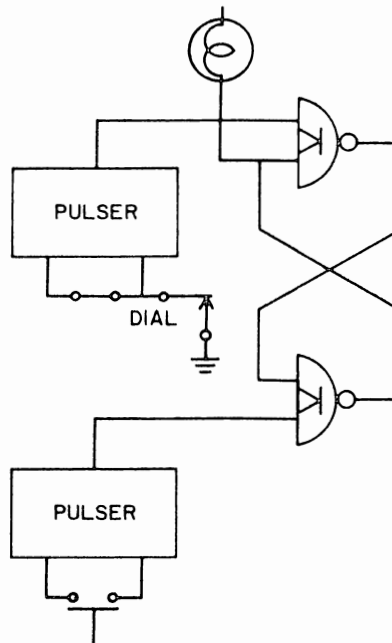


Figure 4 A Flip-Flop with Set and Clear Inputs

Information can also be gated into a flip-flop as shown in Figure 5. Here the flip-flop is set to the ONE state only if the switch is closed when the button is depressed. The toggle switch tells if an action is to take place, and the button tells when the action is to take place. There would be no point in connecting the switch to the flip-flop without the button, as the flip-flop would always go to the ONE state whenever the switch was in the ONE state.

These two elements,

IF a readin is to take place

AND

WHEN the action is to take place

must always be present. In the first example, both elements were present in the same signal. IF the dial was turned, the flip-flop was set IMMEDIATELY. In Figure 5, the switch provided the condition and the button provided the timing.

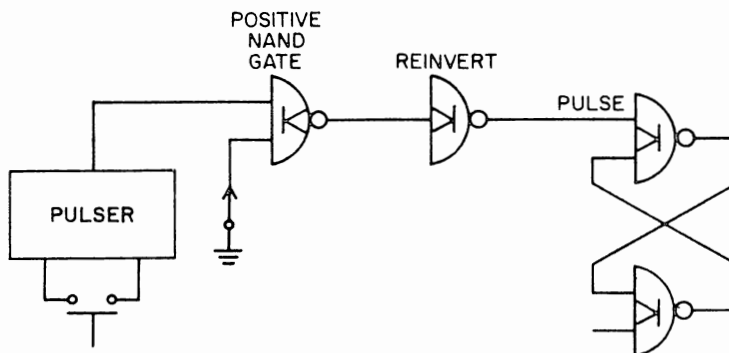


Figure 5 Gating Information into a Flip-Flop

PART 2 DIODE-CAPACITOR-DIODE GATES

In the Logic Laboratory, flip-flop readins normally use a diode-capacitor-diode (DCD) gate. This gate is shown schematically and symbolically in Figure 6. If the level input is held at ground and the pulse input is held at -3 volts, the capacitor will become charged after the setup time has passed. If the pulse input then suddenly goes to ground, a positive going pulse will appear at the output.

The DCD gate forms a 2-input, positive AND gate. A pulse output appears only when the level input AND the pulse input have received the correct signals. Thus, it may be used to gate information into the flip-flop as illustrated in Figures 7 and 8.

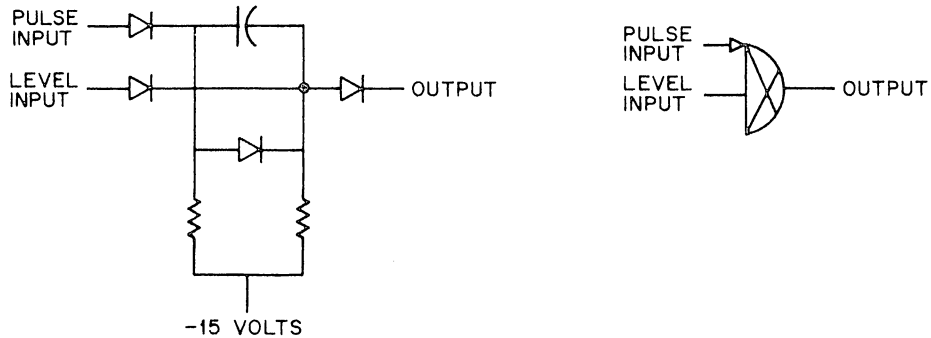


Figure 6 DCD Gate

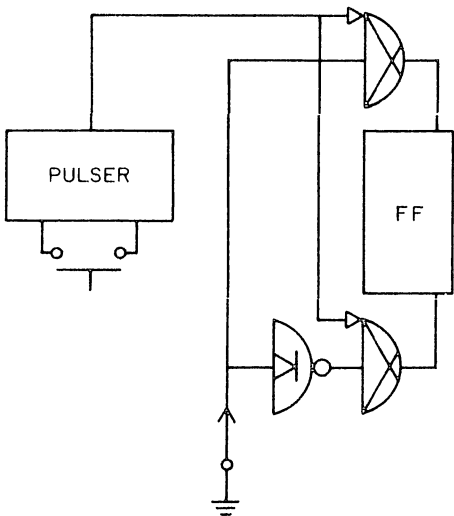


Figure 7 Reading a Toggle Into a FF

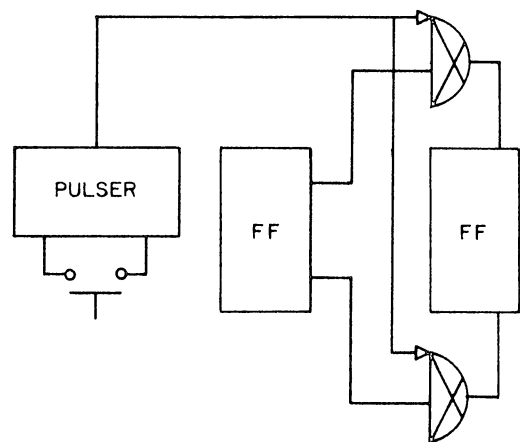


Figure 8 Reading a FF Into a FF

The DCD gate has two other special features which are not a part of a normal AND gate. Because the level input drives an RC circuit with a long time constant, a change in the level input will not begin to affect the circuit until about 100 nsec after the change occurred. Thus, the input has a short delay. A change in the pulse input, on the other hand, produces immediately a short pulse of about 50nsec duration.

Thus, a DCD gate produces an output pulse

WHEN a positive going level change occurs at the pulse input

AND

IF the level input was previously at ground.

These features are useful in swapping information between two (or more) flip-flops, as illustrated in Figure 9. If flip-flop A initially contains a ZERO and flip-flop B initially contains a ONE, a pulse input will interchange the contents of the two flip-flops. The delay and differentiation features of the DCD gates assure that the interchange will occur reliably.

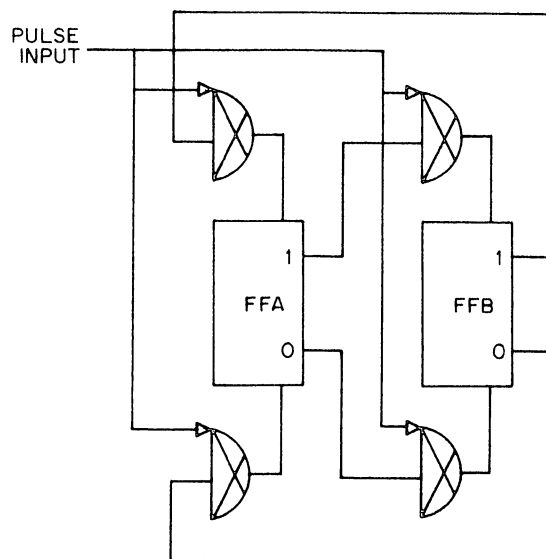
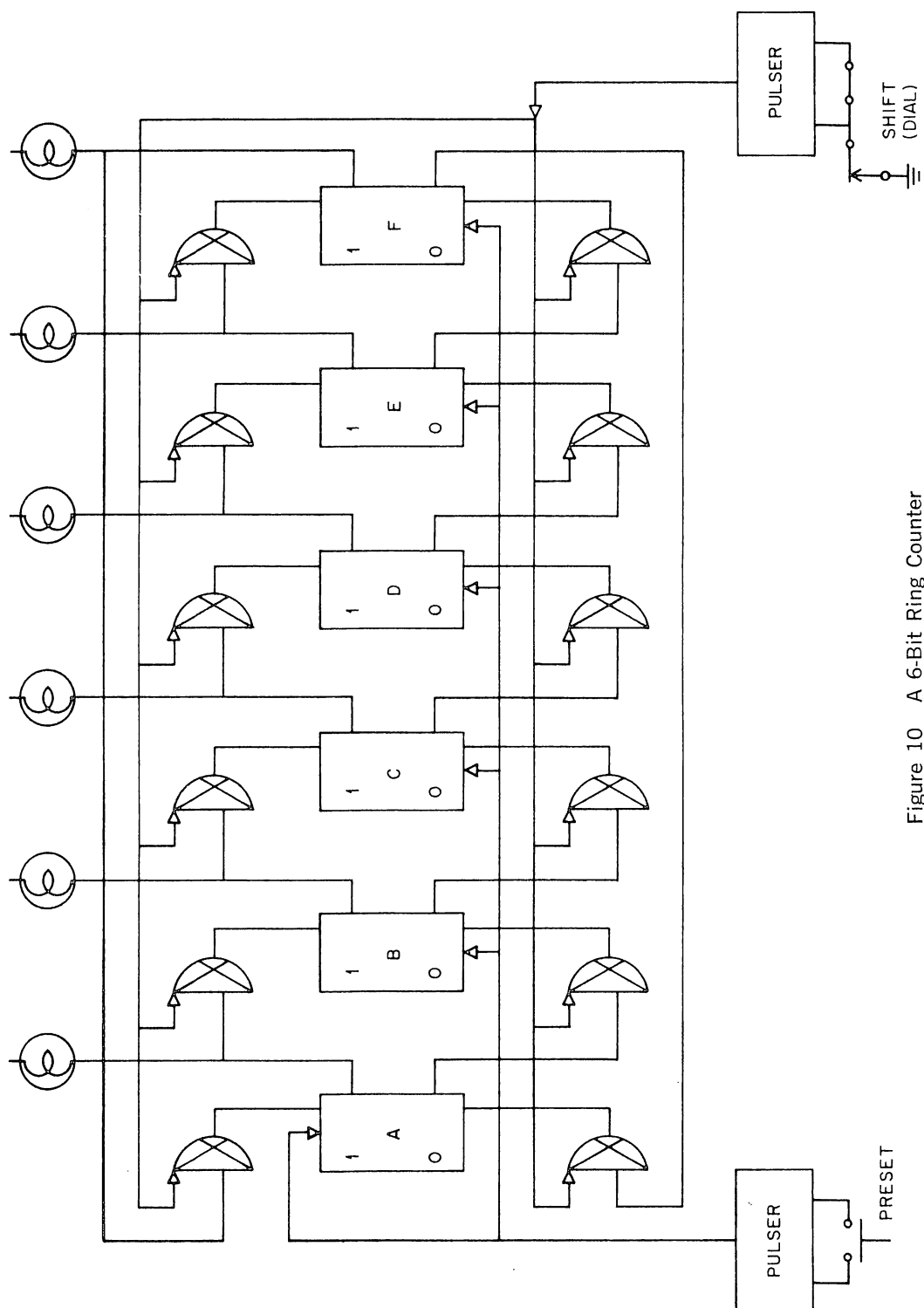


Figure 9 Swapping Information Between Two Flip-Flops

These techniques may be extended to many flip-flops. The result is called a ring counter, because the information will circulate among the entire set. Figure 10 shows the complete circuit for a 6-bit ring counter. The preset line puts the flip-flops in the initial 100000 state. Each shift pulse will then shift the contents of the flip-flops by one position.

2. Make a table showing how the contents of the flip-flops will change with each shift pulse. Construct the circuit and record the results. Do they agree with your theory?

3. The circuit in Figure 10 is called a ring counter because of the way the flip-flops are connected in a loop. A variation, called the switched tail ring counter, reverses the connection from flip-flop F to flip-flop A. That is, the level gate input of the ZERO side of flip-flop A is driven from the ONE output of flip-flop F, and the level gate input on the ONE side of flip-flop A is driven from the ZERO output of flip-flop F. If the switched tail ring counter is initially preset to 100000, what sequence of numbers would you expect at the output? Test your predictions by changing the ring counter to a switched tail ring counter and recording the results.



PART 3 COUNTERS

Sometimes it is desirable to have the state of a flip-flop alternate each time the input is pulsed. This can be done by connecting the flip-flop output terminals to the level inputs, as shown in Figure 11.

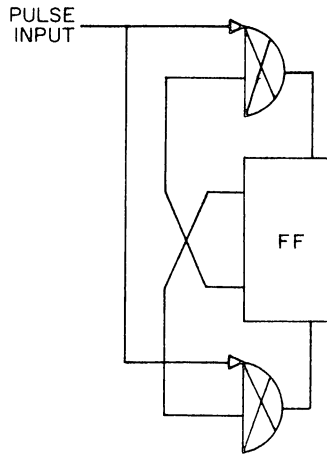


Figure 11 Complement Connection

Thus, if the flip-flop is in the ZERO state, only the ONE input gate is enabled and the first input pulse will change the state of the flip-flop to ONE. This will enable the ZERO input and the next pulse will change the flip-flop back to its original state.

Logically equivalent connections are built into the DCD gates in the Logic Laboratory flip-flops. These use a third gate input as shown in Figure 12. For simplicity, the feedback is not shown on the panel diagram, but it should be remembered because it is a very powerful element of the flip-flop.

4. Understanding now how the flip-flop works, explain in your own words the operation of the counter in Experiment I.

5. Design, build, and test a counter circuit which would operate if the internal feedback to the DCD gates was not built into the flip-flops.

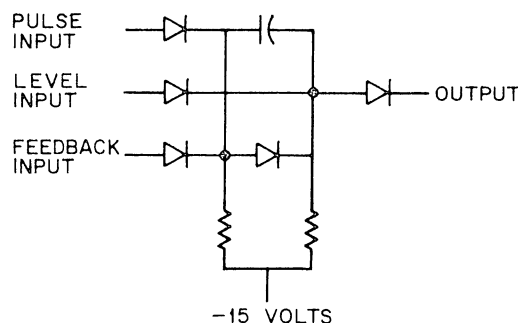


Figure 12 DCD Gate with Feedback Input

PART 4 SPECIAL PROBLEMS

6. What is the function of a flip-flop?
7. What are the two types of information which must be present in order to change the state of a flip-flop?
8. What are the three functions of a DCD gate?
9. Loading is important with flip-flops and DCD gates. Read the Loading Rules Summary in the Appendix and calculate the load for the following:
 - (a) The ZERO output of flip-flop B in Figure 10.
 - (b) The ONE output of flip-flop B in Figure 10.
 - (c) The ZERO output of flip-flop 2 in Figure 7 of Experiment I.
10. Design a 2-bit counter that follows the sequence

00
01
10

then returns to 00. Hint: Use the level inputs on the DCD gates to route every third input pulse so that the counter goes automatically from 10 to 00 instead of 11.

11. Using the same techniques as in problem 10, design, build, and test a circuit that has only five states.

EXPERIMENT V

FROM BOOLEAN EQUATIONS TO GATING NETWORKS

PART 1 REDUCTION TECHNIQUES

In your text you have studied Boolean equations and techniques for minimizing Boolean expressions. In this experiment, you will apply these techniques to build gating networks with a minimum number of circuits.

You will begin with Boolean equations, or a truth table, and reduce this to a minimum Boolean equation. You will then implement the equation with NAND and NOR circuits and reduce the total number of circuits by using the interchangeable polarity convention, DeMorgan's Law, and wired gates.

Going back to the circuit schematic for the Type R121 Diode Gate, you can easily see how DeMorgan's Law works. Figure 1 shows a simple 2-input diode gate. If either A OR B is held at ground, the node point will also go to ground and the output will go negative. On the other hand, only if both A AND B are held negative will the node point go negative and the output go to ground. Thus, the diode gate forms a NOR circuit if the convention is that a ground signal represents a ONE. But, the same circuit forms a NAND gate if the convention is that negative signals represent a ONE.

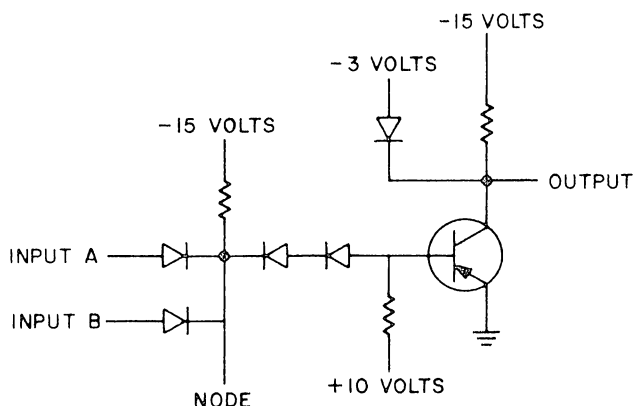


Figure 1 Diode Gate Schematic

In the workbook we will use both conventions. We will indicate which convention we are using by means of diamonds. A hollow diamond \diamond means that ground voltage represents a ONE, while a solid diamond \blacklozenge will be used to indicate that a negative voltage represents a ONE. For example, $\overline{A} \diamond$ means that the voltage will be at ground when A is a ONE, while $\overline{A'} \blacklozenge$ means that the voltage will be at ground when A' is a ONE.

In this way, the convention may be changed at any point in the logic diagram. Thus, the same diode gate may be used to form an AND or an OR circuit as well as a NAND or a NOR circuit. Figure 2 shows how this convention works.

The logical functions are also shown in the corner of the symbol. This is not necessary, since the logic function can be derived from the hardware symbol and the diamonds, but it will be used here to make the drawings easier to read.

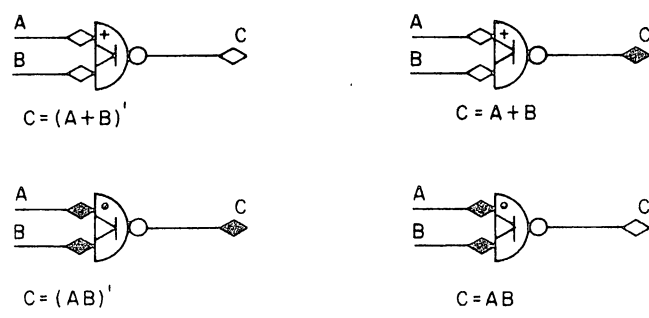


Figure 2 Logic Convention for R121

These same techniques can be applied to a Type R122 Diode Gate. It also can form an AND gate, an OR gate, a NAND gate, or a NOR gate. Figure 3 shows the logic convention.

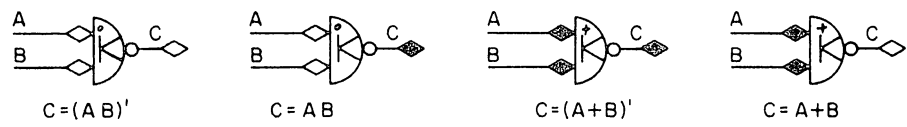


Figure 3 Logic Convention for R122

In some cases, the diode gates are not necessary at all. The logical function may be performed simply by the tying together of the appropriate wires. If the input signals already come from diode gates or pulsers, the additional diodes may frequently be eliminated.

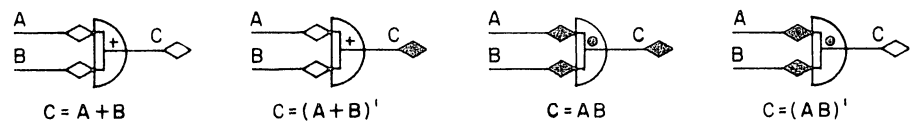


Figure 4 Wired Gates

Figure 4 shows how such a wired gate would operate. Lines A and B come from diode gate outputs. If either line is held at ground by a saturated transistor, the entire line will be grounded. Only if neither A nor B is grounded will the line be allowed to go negative.

Tying the lines together means that the signals A and B are no longer independent, only the combined signal $A + B$ or AB is available. Therefore, this technique cannot be used if either A or B is required for some other activity. (As for example, if you wish to light one indicator on the condition $A + B$ and another on the condition A only.) This technique also cannot be used with flip-flops since the action of the other signals could force the state of the flip-flop.

To understand how the interchangeable polarity convention, DeMorgan's Law, and

wired gates simplify the implementations of a logical expression, consider the expression

$$AB' + A'B$$

This is a commonly used function, called the exclusive OR and written as:

$$A \oplus B$$

Using only the convention that a positive voltage equals a ONE and the most straightforward logic, the exclusive OR function requires eight gates. This circuit is shown in Figure 5.

However, if the lines are labeled as shown in Figure 6, it becomes clear that many of the gates which perform only inversion are not required. We may simply eliminate gate 8. Also, gates 5 and 6 may be removed if we replace the positive OR gate (7) with a negative OR gate. This produces the circuit of Figure 7, which requires only five gates instead of eight.

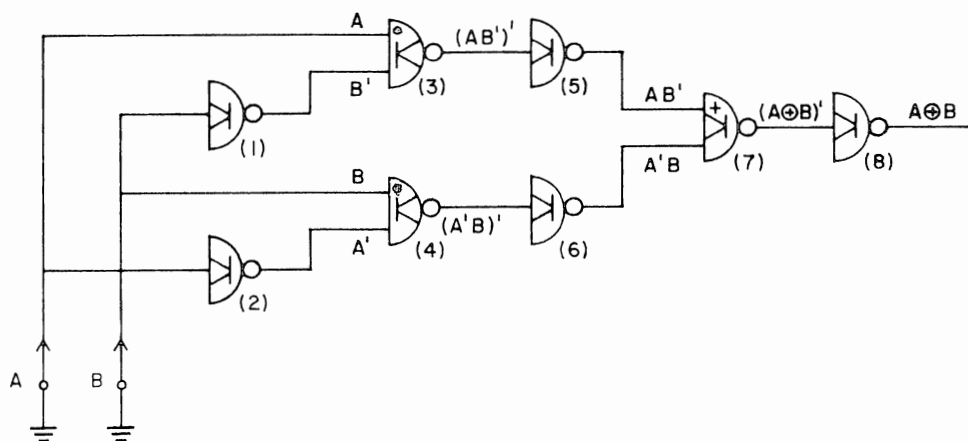


Figure 5 $A \oplus B$, Single Polarity Convention

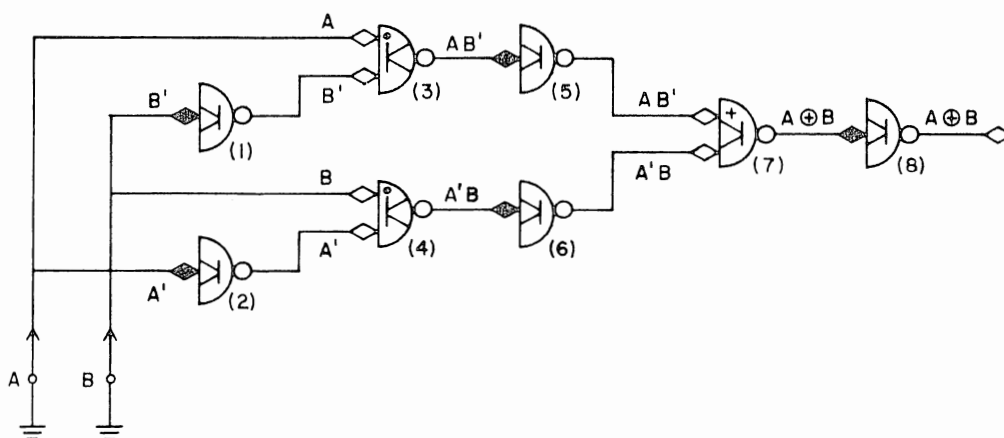


Figure 6 $A \oplus B$, with Interchangeable Polarity Convention

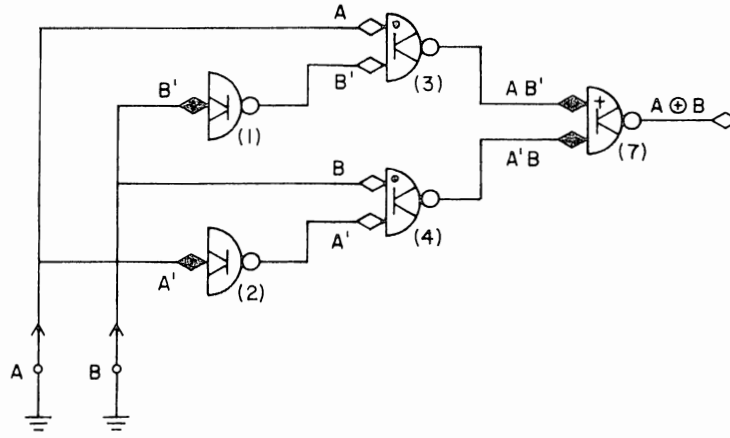


Figure 7 $A \oplus B$ Simplified

Consider gates 1 and 2 which perform only inversions. To see if any of these can be eliminated let us apply DeMorgan's Law to the original expression:

$$\begin{aligned}
 A \oplus B &= AB' + A'B \\
 (A \oplus B)' &= (A' + B)(A + B') \\
 &= A'A + A'B' + BA + BB' \\
 &= A'B' + AB
 \end{aligned}$$

thus:

$$A \oplus B = (A'B' + AB)'$$

and since our inputs are ground for a ONE and negative for a ZERO, the term AB can be formed with a positive AND and the term $A'B'$ can be formed with a negative AND. This is shown in Figure 8.

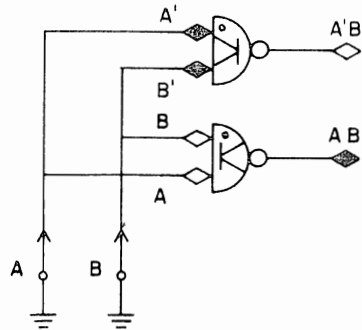


Figure 8 $A'B'$; AB

These are of opposite polarities, so one must still be inverted before they can be ORed together. However, one less gate is required for inversion than was used previously. The total circuit is shown in Figure 9.

Now consider the possibility of using a wired gate. A physical gate must be used to form the function AB since line A drives two inputs and must therefore be independent of line B . The same is true of the gate that forms $A'B'$. However, the lines AB and $A'B'$ do not need to be independent. A wired gate can be used to OR these signals together, yielding the form of Figure 10. This is the simplest form, requiring only three gates instead of the original eight.

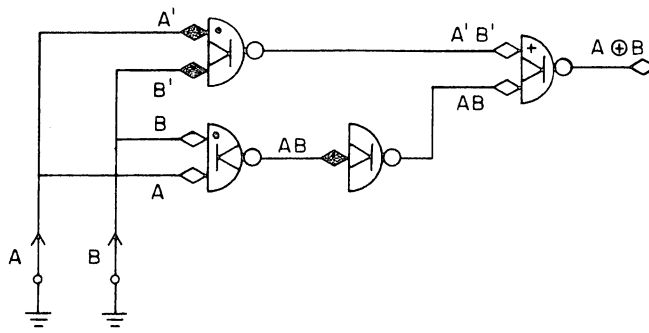


Figure 9 $A \oplus B$

Simplified with Interchangeable Polarity Convention and DeMorgan's Law

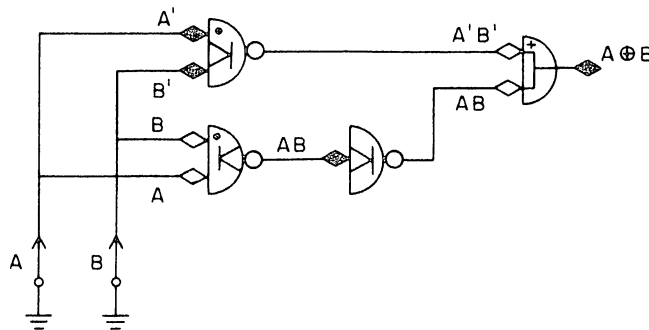


Figure 10 $A \oplus B$, Simplest Form

When more than one expression is to be implemented, the circuitry can be minimized by forming common terms only once. For example, the following two expressions:

$$A \oplus B$$

$$(A \oplus B)C'$$

require only four gates. This is illustrated in Figure 11.

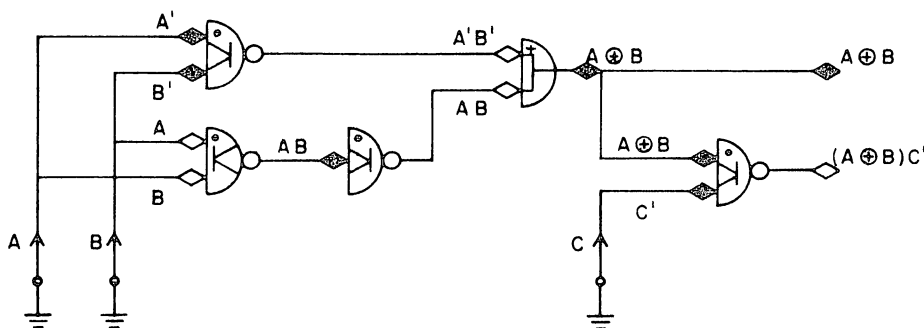


Figure 11 Two Equations with a Common Term

PART 2 PROBLEMS

1. Draw a truth table for the circuit of Figure 12. Simplify the circuit and test it to be sure that the same truth table is obtained. Record both truth tables.

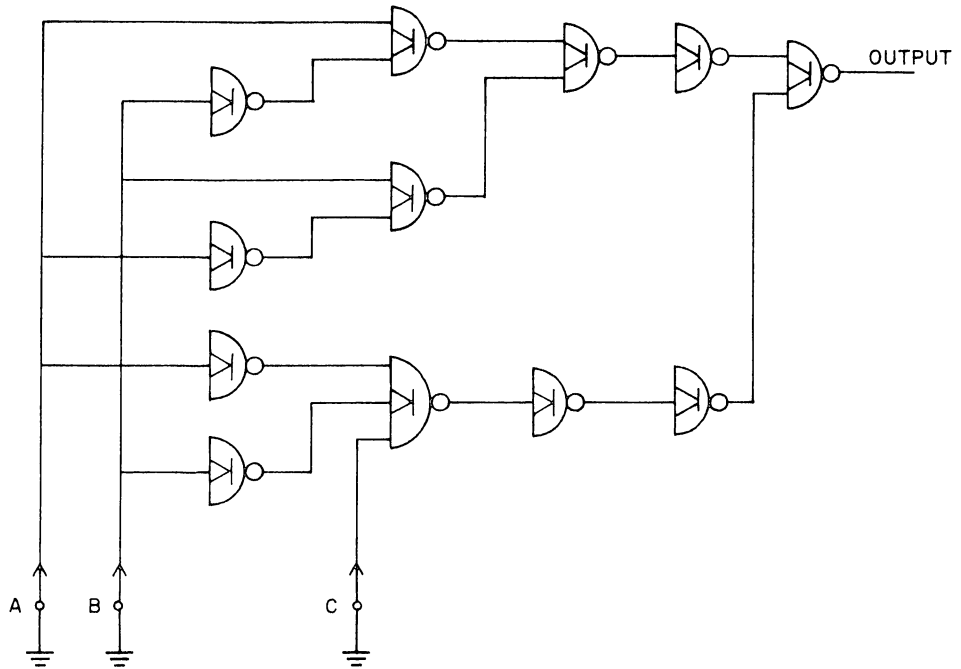


Figure 12 Gating Network

2. Implement the following equations. Reduce them to a minimum number of circuits. Test them and record the truth table.

- (a) $C(AB' + A'B) + AB$
 (b) $A \oplus (B \oplus C)$
 (c) $A + A'BC$
 (d) $C(AB' + A'B) + AB; \quad A'B' + AB$
 (e) $A \oplus B; \quad (A \oplus B) + C$

3. Implement the following truth table. Reduce it to a minimum number of circuits. Write down your original and final expression and test your circuit.

A	B	C	Function
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

4. Implement one of the gating networks which you designed for a homework problem. Reduce it to a minimum number of circuits. Build it. Test it and record your results.

EXPERIMENT VI

BOOLEAN EQUATIONS AND FLIP-FLOPS

PART 1 DUALITY

Just by changing definitions, a NOR gate can become a NAND, OR, or AND gate. These same definition changes can be applied to any digital circuit element. YOU define the state of the element in the manner most convenient to you—in the way that minimizes the circuitry.

Consider the problem of lighting an indicator when either of two switches is in the ONE state. Figure 1 shows how simple this circuit can be if we define the open position as the ONE state. If we make the opposite definition, the circuit will require an extra gate to light the indicator on the OR condition and still keep lines A and B independent. Try it.

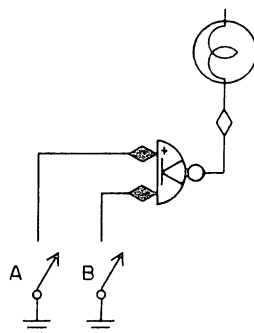


Figure 1 Lighting an Indicator

Clearly, changes in definition should not be made arbitrarily as they will only confuse the person who must use your equipment. Thus, you would not want one switch to be open for a ONE while another is closed for a ONE. And when such changes are made, they should always be labeled in such a way that anyone who is reading your drawing can tell what you are doing. In drawing the switches, we have shown each switch in the position that corresponds to a logical ONE.

Definition changes are frequently used for flip-flops. So far we have considered the flip-flop as a device with two outputs. Truly, there are only two outputs, but logically we may consider them in four ways. These are

- the output which is ground when the flip-flop holds a ONE,
- the output which is negative when the flip-flop holds a ONE,
- the output which is ground when the flip-flop holds a ZERO,
- and
- the output which is negative when the flip-flop holds a ZERO.

Figure 2 illustrates how these terminals are shown on a drawing.

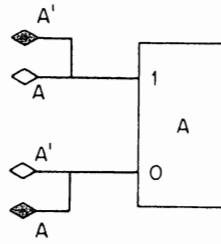


Figure 2 Four Flip-Flop Outputs

These four logical outputs can make gating circuits much simpler. Consider the exclusive OR circuit. When the input comes from switches, as in Figure 3, three gates are required. Figure 4 shows the same circuit with the inputs being driven by flip-flops. In this case the “negative” outputs are used and only two gates are required.

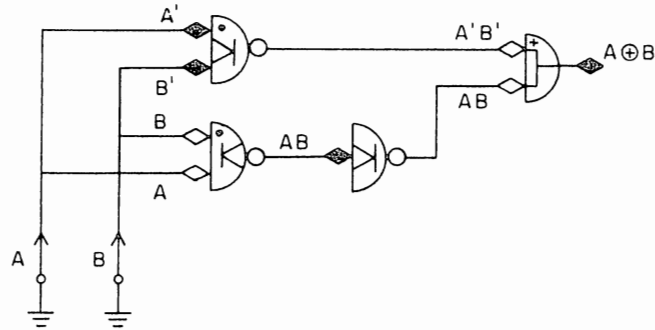


Figure 3 Exclusive OR with Switch Inputs

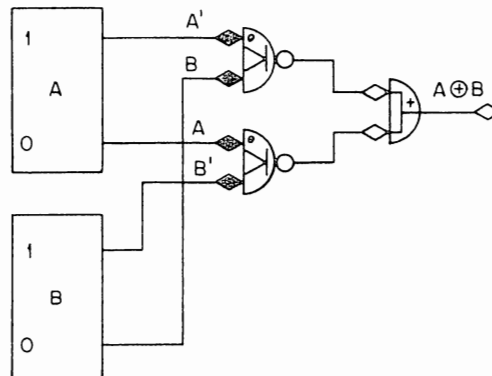


Figure 4 Exclusive OR with Flip-Flop Inputs

1. In Experiment V you implemented Boolean expressions using switches for inputs. Now, using flip-flops to provide the inputs, redesign the circuitry. With the four output notation, try to minimize the number of gates. Compare your new results with those of Experiment V.

- (a) $C(AB' + A'B) + AB$
- (b) $A \oplus (B \oplus C)$
- (c) $A + A'BC$
- (d) $C(AB' + A'B) + AB; \quad A'B' + AB$
- (e) $A \oplus B; \quad (A \oplus B) + C$

2. Construct and test the circuits which you designed for problem 1. In order to test the circuits, it is necessary to provide an input to the flip-flops. If you connect the flip-flops in a counter configuration, with a pulser as the input, you can step through each of the states quite easily. Figure 5 shows how the exclusive OR circuit would be set up for a test.

3. Using the techniques described in questions 1 and 2, design, construct, and test a binary-to-octal decoder. Use three flip-flops to hold the binary number and eight lights to indicate the output. Make use of the four output terminal notation to keep the number of gates to a minimum.

PART 2 TIMING CONSIDERATIONS

Because DCD gates have delay and flip-flops have memory, the state of a flip-flop at any given time will depend on the state of the inputs at some previous time. The time will be represented here as a function of t and will always be enclosed in parentheses, for example: $(t + 1)$, (t) , $(t0)$, $(t1)$, etc. The time may be written after each variable as: $A(t0) \oplus B(t0)$ or, after an entire function as $A \oplus B|_{(t0)}$.

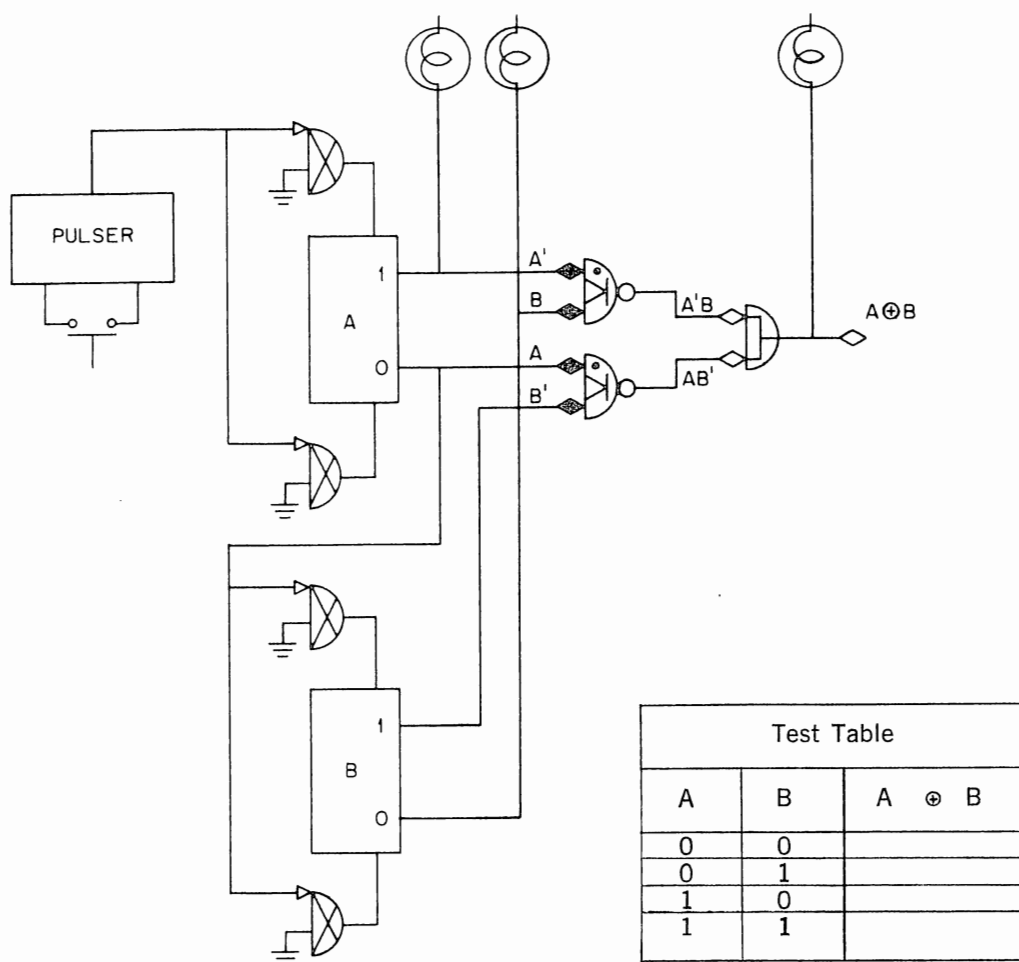


Figure 5 Exclusive OR Test Set Up

The need for a time representation may be seen from the simple ring counter circuit shown in Figure 6. If the flip-flops are in any arbitrary state, specified as $A(t)$, $B(t)$, $C(t)$, $D(t)$, $E(t)$, and $F(t)$, and if a pulse arrives one unit of time later, the flip-flop states will be changed as specified in the following equations:

$$A(t + 1) = F(t)$$

$$B(t + 1) = A(t)$$

$$C(t + 1) = B(t)$$

$$D(t + 1) = C(t)$$

$$E(t + 1) = D(t)$$

$$F(t + 1) = E(t)$$

PART 3 INPUT EQUATIONS

The R201 flip-flop has five gates, two on the set side and three on the clear side. Since these gates are preconditioned by the flip-flop itself, the final flip-flop state depends on the previous state of the flip-flop itself, as well as the states of the inputs.

To write an equation for the state of a flip-flop, we must ask what conditions will result in the flip-flop holding a ONE and what conditions will result in the flip-flop holding a ZERO. There are two possibilities which will result in a ONE: if the flip-flop was in the ZERO state and it received a set input; or if it was a ONE and it did not receive a clear signal. If neither of these conditions is true, the flip-flop will go to, or remain in, the ZERO state. These conditions are expressed in the table below.

A(t)	Set Rec.	Clear Rec.	A(t+1)
0	yes	—	1
1	—	no	1
0	no	—	0
1	—	yes	0

If the inputs are labeled as in Figure 7, we can write an equation for the state of the flip-flop at time $t + 1$ as a function of the variables A , B , C , D , E , and F , at time t . That is:

$$A(t + 1) = A'(B + D) + A(C + E + F)' \Big|_{(t)}$$

which will simplify to:

$$A(t + 1) = A'B + A'D + AC'E'F' \Big|_{(t)}$$

From this equation, we may consider the special cases. For example, if only two gates are used, as in Figure 8a, then

$$D = E = F = 0$$

$$D' = E' = F' = 1$$

and the equation reduces to

$$A(t + 1) = A'B + AC' \Big|_{(t)}$$

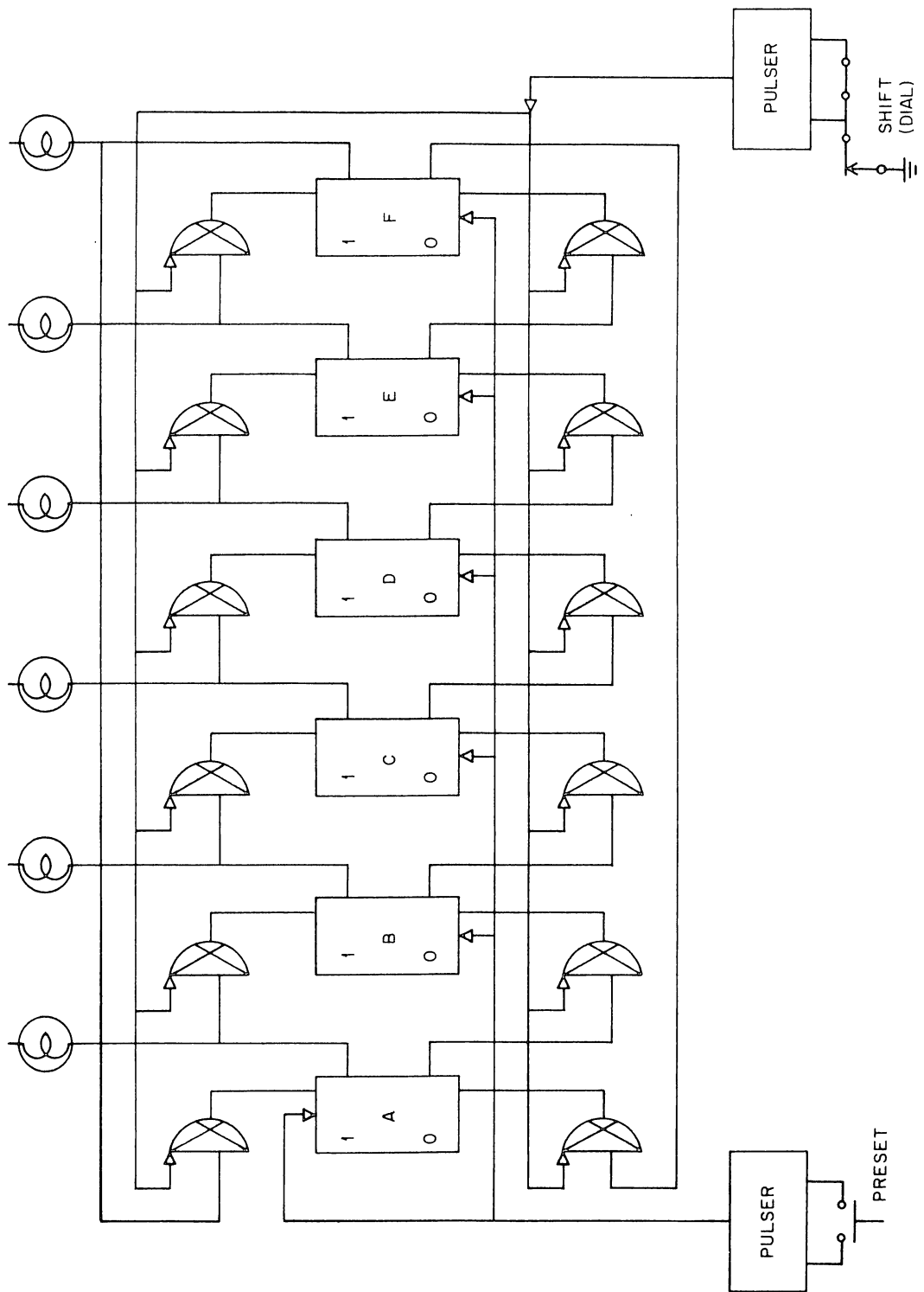
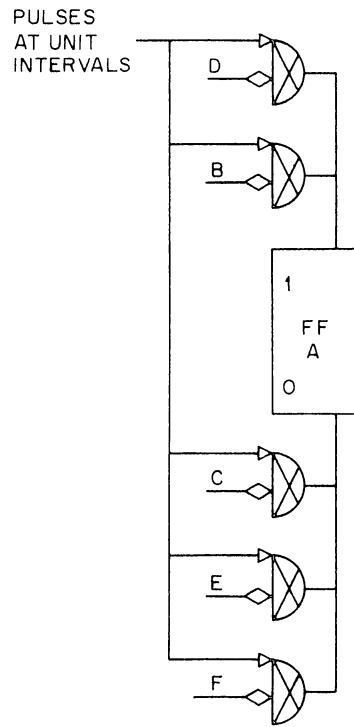


Figure 6 A 6-Bit Ring Counter



$$A(t+1) = A'(B+D) + A(C+E+F)'$$

$$A(t+1) = A'B + A'D + AC'E'F'$$

Figure 7 Flip-Flop State

In the case of a jam transfer input, as in Figure 8b, $C = B'$ and

$$A(t+1) = A'B + AB \Big|_{(t)} = B(t)$$

Thus, flip-flop A is jammed to the value of B, independent of its own previous contents.

In the case of a conditional complement input, Figure 8e, $C = B$ and the equation reduces to

$$A(t+1) = A'B + AB' \Big|_{(t)} = A(t) \oplus B(t),$$

the exclusive OR function. Figures 8a–8f show these and other special cases and the equations to which they reduce.

The most commonly used input configurations are the jam transfer, the complement, and a two-step process—an unconditional clear followed by a conditional set. However, in some cases the input configuration is quite complex. It is in these cases that the Boolean expressions are most valuable.

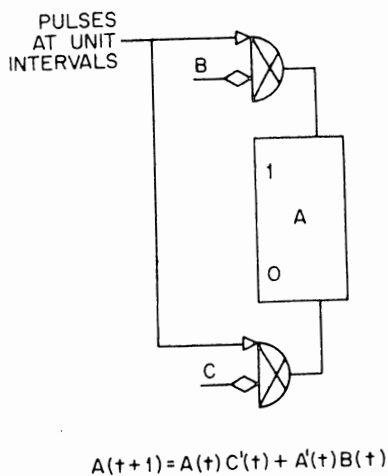


Figure 8a Set and Clear Gated Separately

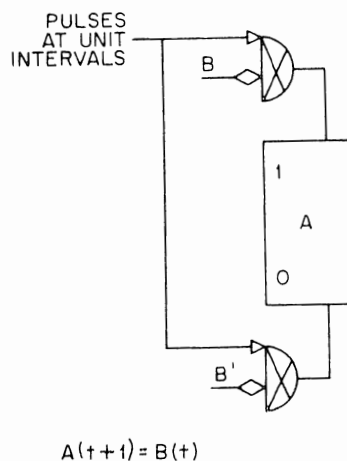


Figure 8b Jam Transfer

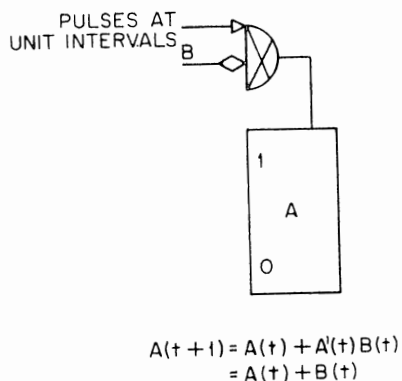


Figure 8c Set Input Only

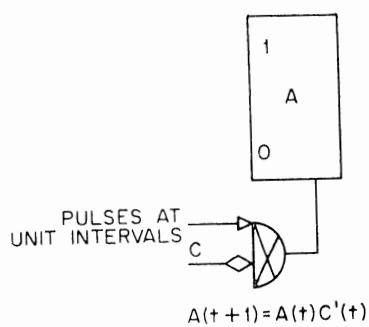


Figure 8d Clear Input Only

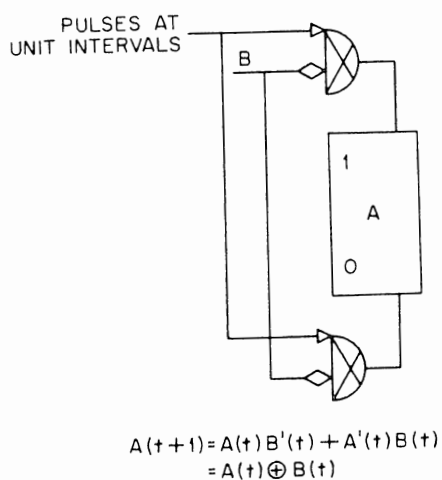


Figure 8e Conditional Complement

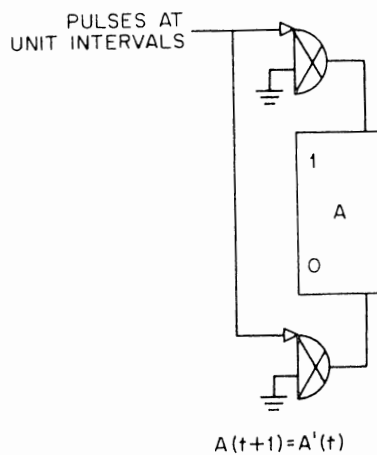


Figure 8f Complement

Figure 9 shows a typical circuit where most inputs are the simple jam transfer type, but the input to the one flip-flop is a complex function.

This circuit is a maximal length, pseudo random number generator. In other words, it uses 15 of the 16 possible states, but its output is gibberish when interpreted in the binary number system. Such circuits are frequently used in studying random events. Although the output looks random, the circuitry can be reproduced at any time and the exact same sequence of "pseudo random" events will be repeated.

The equations for this circuit are:

$$\begin{aligned} A(t+1) &= C(t) \oplus D(t) + A'(t)B'(t)C'(t)D'(t) \\ B(t+1) &= A(t) \\ C(t+1) &= B(t) \\ D(t+1) &= C(t) \end{aligned}$$

4. Construct the maximal length, pseudo random number generator and test it. No matter what state the flip-flops are in, this circuit will generate a repetitive pattern of 15 states.

- (a) From the equations, determine what this pattern is.
- (b) Test the circuit to determine the pattern.

5. Construct and test circuits which operate according to the following equations. Try to minimize the amount of circuitry involved.

$$\begin{aligned} \text{(a) } A(t+1) &= B \oplus C \oplus D + AB'CD' \\ B(t+1) &= A \\ C(t+1) &= B \\ D(t+1) &= C \end{aligned} \quad \begin{aligned} \text{(b) } A(t+1) &= D' \\ B(t+1) &= A + B'C'D' \\ C(t+1) &= B \\ D(t+1) &= C \end{aligned}$$

PART 4 PULSE INPUTS

In some cases, information is also carried in the pulses. These signals may be treated just as level gate inputs, except that the pulse often carries more information. If the pulse input comes from a flip-flop output, for example, it tells the state of the flip-flop output at time t and at time $t + 1$.

A typical example of this is the counter circuit shown in Figure 10. The input to flip-flop F is from the pulser, so the equation may be written simply as

$$F(t+1) = F'(t).$$

A positive going transition at the ZERO output of flip-flop F tells that $F'(t+1)F(t) = 1$. When this drives the two inputs to flip-flop E , the equation for flip-flop E is

$$E(t+1) = E(t) \oplus [F'(t+1)F(t)].$$

6. Write the equations of flip-flops A , B , C , and D in the counter.

7. Using these techniques and those described in Parts 3 and 4 of Experiment IV, design and build a counter which has nine states.

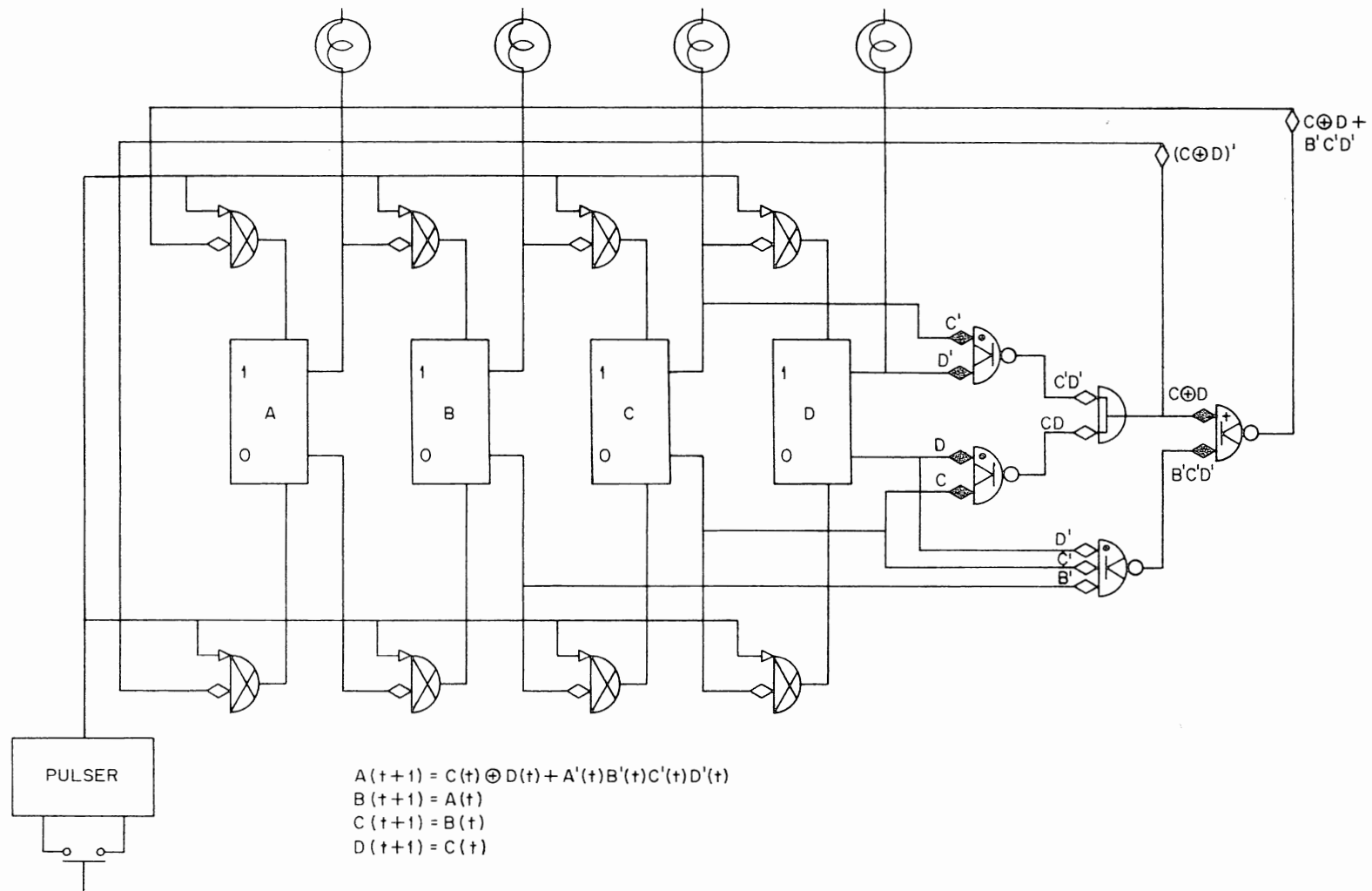


Figure 9 Maximal Length, Pseudo Random Number Generator

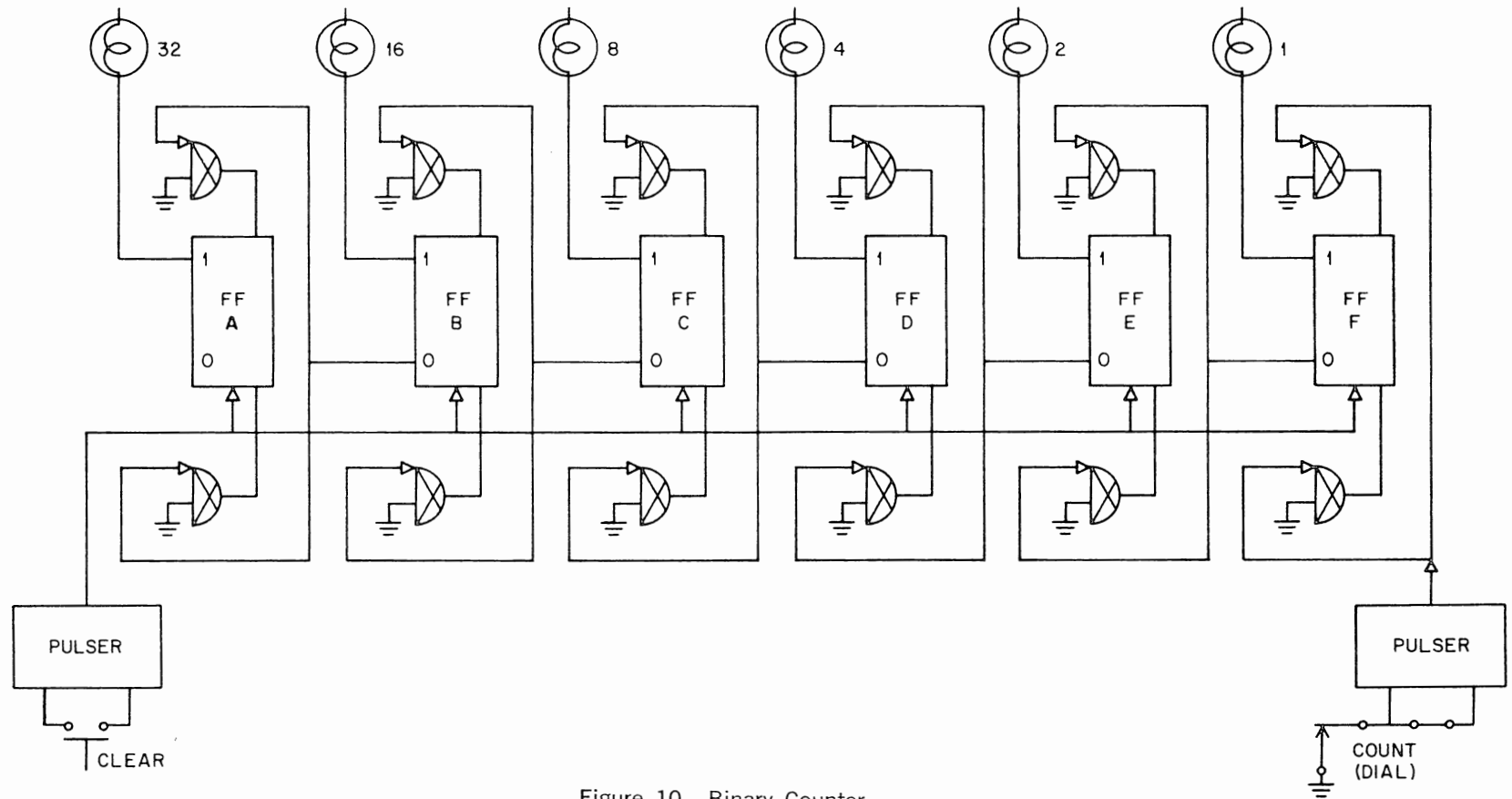


Figure 10 Binary Counter

EXPERIMENT VII

ADDITION

PART 1 TECHNIQUES OF ADDITION

Using simple 2-state elements and manipulating them with Boolean algebra, we may perform sophisticated operations such as addition. How? By representing the numbers in the binary system and adding them in a manner similar to that used with decimal numbers. That is:

$$\begin{aligned}0 \text{ plus } 0 &= 0 \\0 \text{ plus } 1 &= 1 \\1 \text{ plus } 0 &= 1\end{aligned}$$

and since there is no binary symbol for 2,

$$1 \text{ plus } 1 = 0 \text{ with } 1 \text{ to carry.}$$

Because there may be a carry of 1 from any bit to the next, the addition rules must also provide for a carry input.

$$\begin{aligned}1 \text{ plus } 0 \text{ plus } 0 &= 1 \\1 \text{ plus } 0 \text{ plus } 1 &= 0 \text{ with } 1 \text{ to carry} \\1 \text{ plus } 1 \text{ plus } 0 &= 0 \text{ with } 1 \text{ to carry} \\1 \text{ plus } 1 \text{ plus } 1 &= 1 \text{ with } 1 \text{ to carry.}\end{aligned}$$

Since the sum of two binary bits plus a carry never produces a carry greater than 1, we have considered all the cases. A truth table, as below, expresses all the conditions very concisely.

Inputs			Outputs	
Carry	A	B	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

From this truth table we can write the following equations:

$$\begin{aligned}S &= C'A'B + C'AB' + CA'B' + CAB \\C(\text{out}) &= C'AB + CA'B + CAB' + CAB\end{aligned}$$

representing the addition process for a single pair of binary bits.

Since complete numbers contain many binary bits, the addition equations must be applied to each pair. This may be done by loading the augend and addend into two registers and shifting the numbers serially through the adder logic. Or, the adder logic may be repeated for each pair of binary bits, so that the complete numbers are added in parallel. The choice of whether to build a serial adder or a parallel adder depends on the required speed, the available equipment (or funds), and the format of the inputs and outputs. Serial addition is the less expensive technique, since only one adder circuit is required. Parallel addition normally is faster. However, if the numbers arrive or are sent out in a serial fashion, serial addition can be performed equally fast.

PART 2 SERIAL ADDERS

Serial addition requires two shift registers (to hold the augend and addend) and an adder circuit, as illustrated in Figure 1. It is not necessary to have a separate register for the sum. As each pair of bits are added together, they may be discarded and one of the flip-flops may be used to store the resultant sum bit. Since the carry must be stored between shift pulses, one extra flip-flop is added for that purpose.

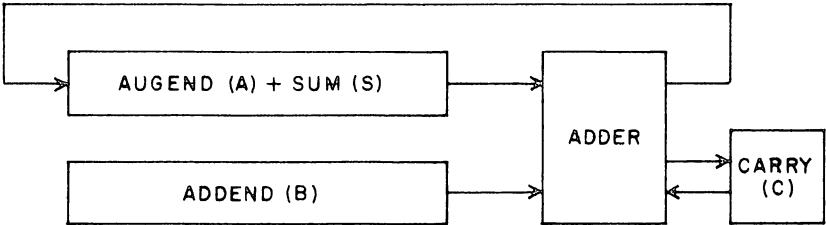
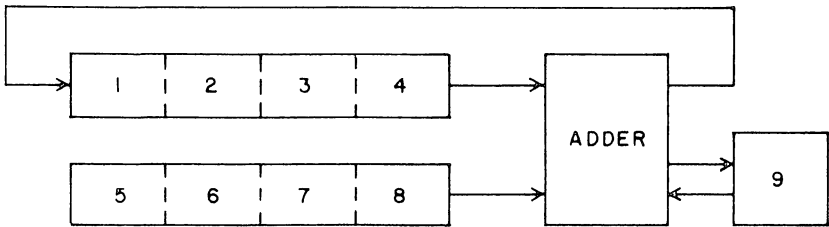


Figure 1 Block Diagram of a Serial Adder

Figure 2 shows the sequence of events which takes place in a 4-bit serial adder. A is the augend, B is the addend, C is the carry, and S is the sum. A3, B3, and S3 are the least significant bits of the augend, addend, and sum respectively. C3 is a carry that is generated by adding A3 and B3.



Time	Action	Flip-Flops								
		1	2	3	4	5	6	7	8	9
0	CLEAR	0	0	0	0	0	0	0	0	0
1	READ	A0	A1	A2	A3	B0	B1	B2	B3	0
2	SHIFT	S3	A0	A1	A2	—	B0	B1	B2	C3
3	SHIFT	S2	S3	A0	A1	—	—	B0	B1	C2
4	SHIFT	S1	S2	S3	A0	—	—	—	B0	C1
5	SHIFT	S0	S1	S2	S3	—	—	—	—	C0

Figure 2 Contents of Flip-Flops

Figure 3 shows the logic of the adder and carry circuits. The sum is generated exactly as shown in the equation of Part 1. The carry logic is simplified somewhat to take advantage of the fact that the carry flip-flop always remains the same except when A AND B are both ZERO, or when A AND B are both ONE.

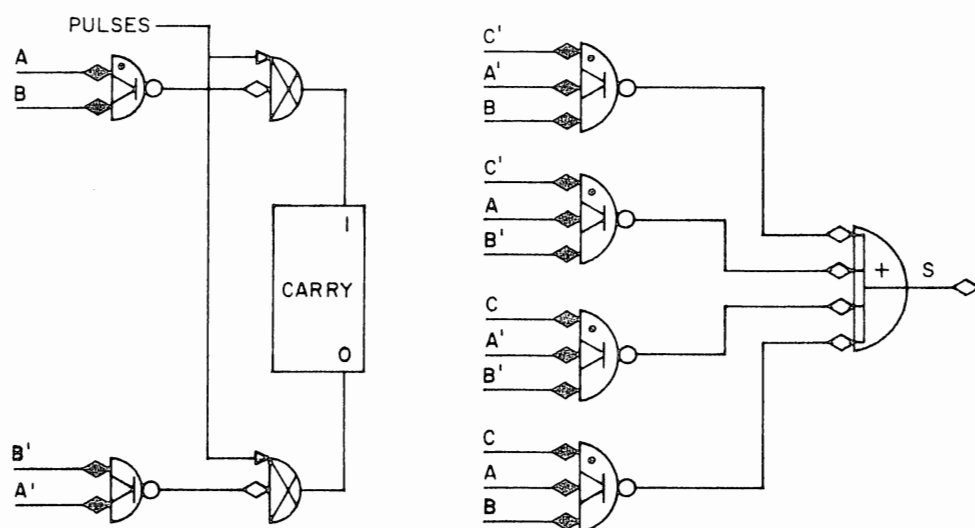
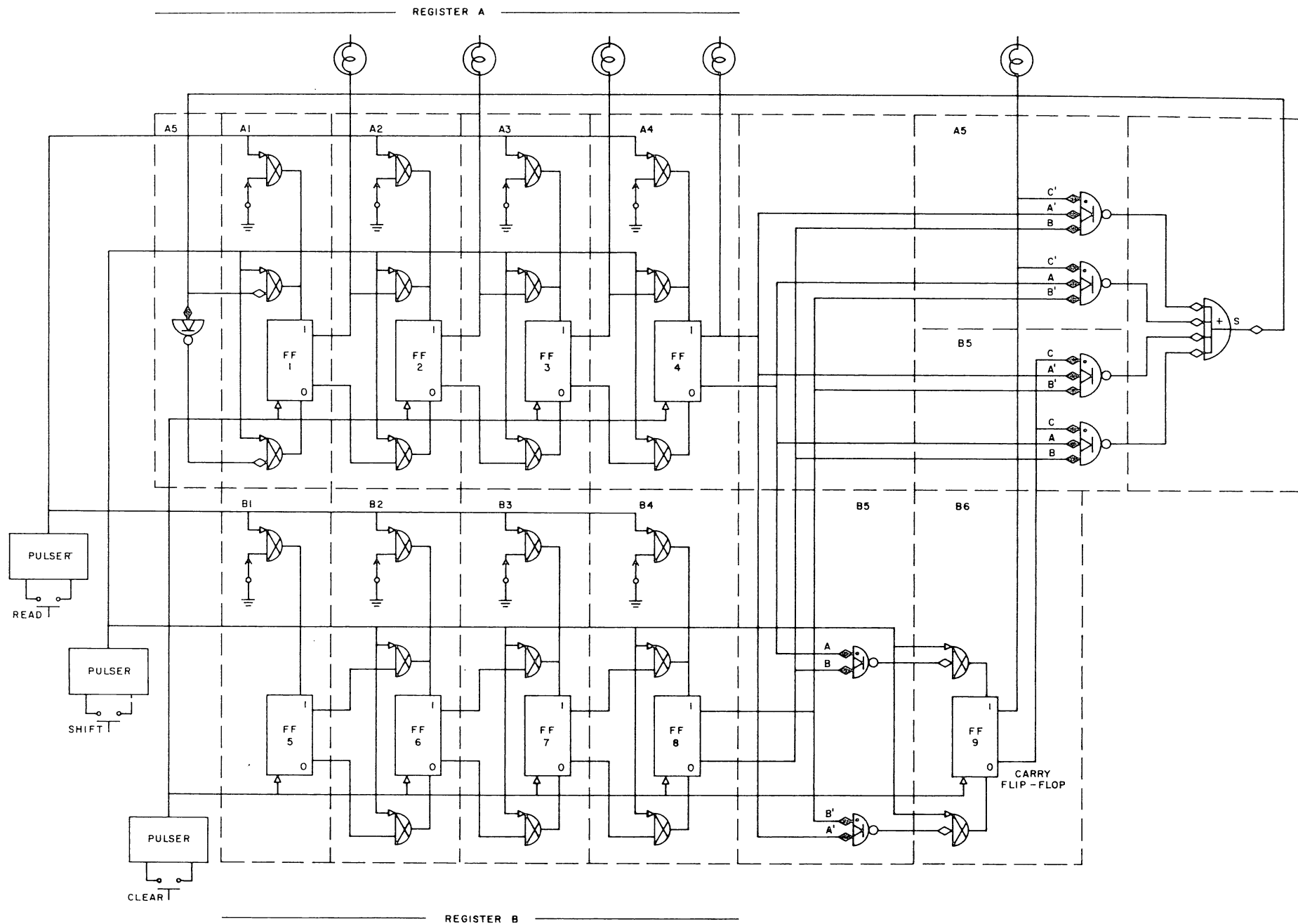


Figure 3 Adder Logic

Figure 4 shows a complete adder with inputs from the toggle switches, and three push buttons which “clear” the adder, “read” in the numbers from the toggle switches, and “shift” and add. The modules are separated with dotted lines, and the location is shown with an alphanumeric symbol. A indicates the top logic rack; B, the next lower rack. The individual positions are numbered 0–9 from the left.

1. How many shift pulses are required to complete an addition?
2. What is the sequence of operations which must be performed for a complete addition?
3. If register A contains the number 0101 and register B contains the number 0111, what numbers will register A and the carry flip-flop contain at the end of each step in the addition?
4. If the number A is 0010 and the number B is 1010, what will register A and the carry flip-flops contain at the completion of each step?
5. Construct the counter as illustrated in Figure 4. To make wiring easier, cross off each wire on the diagram as you put it in. From your answers to questions 3 and 4, construct a test table and check the wiring.



6. What are the sums of the following numbers?

- | | |
|--------------|--------------|
| (a) 3 plus 3 | (c) 7 plus 2 |
| (b) 9 plus 5 | (d) 9 plus 8 |

7. Add the following numbers and draw a diagram showing the contents of register A and the carry flip-flop at each step during the addition process.

- | |
|---------------|
| (a) 7 plus 3 |
| (b) 7 plus 12 |

PART 3 A SERIAL BINARY SUBTRACTOR

Binary subtraction may be performed in a manner quite similar to binary addition. The possible cases are:

$$\begin{aligned}0 - 0 &= 0 \\0 - 1 &= 1 \text{ with 1 to borrow} \\1 - 0 &= 1 \\1 - 1 &= 0\end{aligned}$$

and with a borrow input (-1):

$$\begin{aligned}-1 + 0 - 0 &= 1 \text{ with 1 to borrow} \\-1 + 0 - 1 &= 0 \text{ with 1 to borrow} \\-1 + 1 - 0 &= 0 \\-1 + 1 - 1 &= 1 \text{ with 1 to borrow}\end{aligned}$$

Since the borrow out never exceeds 1, these equations cover all the conditions for the subtraction of one bit from another.

8. From the subtraction equations, construct a truth table showing the borrow in, minuend, subtrahend, difference, and borrow out. Use the same format as in the adder truth table and note the similarity between the sum and the difference.

9. Write equations for the difference and the borrow out.

10. Modify the adder circuit to form a binary subtractor. Store the minuend in register A and the subtrahend in register B. Use the carry flip-flop to store the borrow.

11. Prepare a table, similar to that of Figure 2, showing the contents of the individual flip-flops at each step.

12. Prepare a test table by calculating the sequence of events for the subtractions below. Then test your design by performing the subtractions and recording the results.

- | |
|---------------|
| (a) 9 minus 5 |
| (b) 8 minus 2 |

EXPERIMENT VIII

PARALLEL ADDITION AND SUBTRACTION

PART 1 THE TWO-STEP PARALLEL ADDER

Parallel logic allows high speed addition of long binary numbers. The parallel addition process requires only a small number of steps regardless of the size of the augend and addend.

However, the basic logic is repeated so many times in a parallel adder that it is extremely important to minimize the circuitry used in each individual adder. For this reason, a compromise technique is frequently used—the addition is performed in two steps. The first step, half add, forms the exclusive OR of the augend and addend, while the second step produces and adds in all of the carries.

Figure 1 shows a diagram of a single pair of bits in a 2-step adder. The augend bit is in flip-flop A and the addend bit is in the toggle switch. The addition equations below the diagram have been rewritten to show how this logic operates.

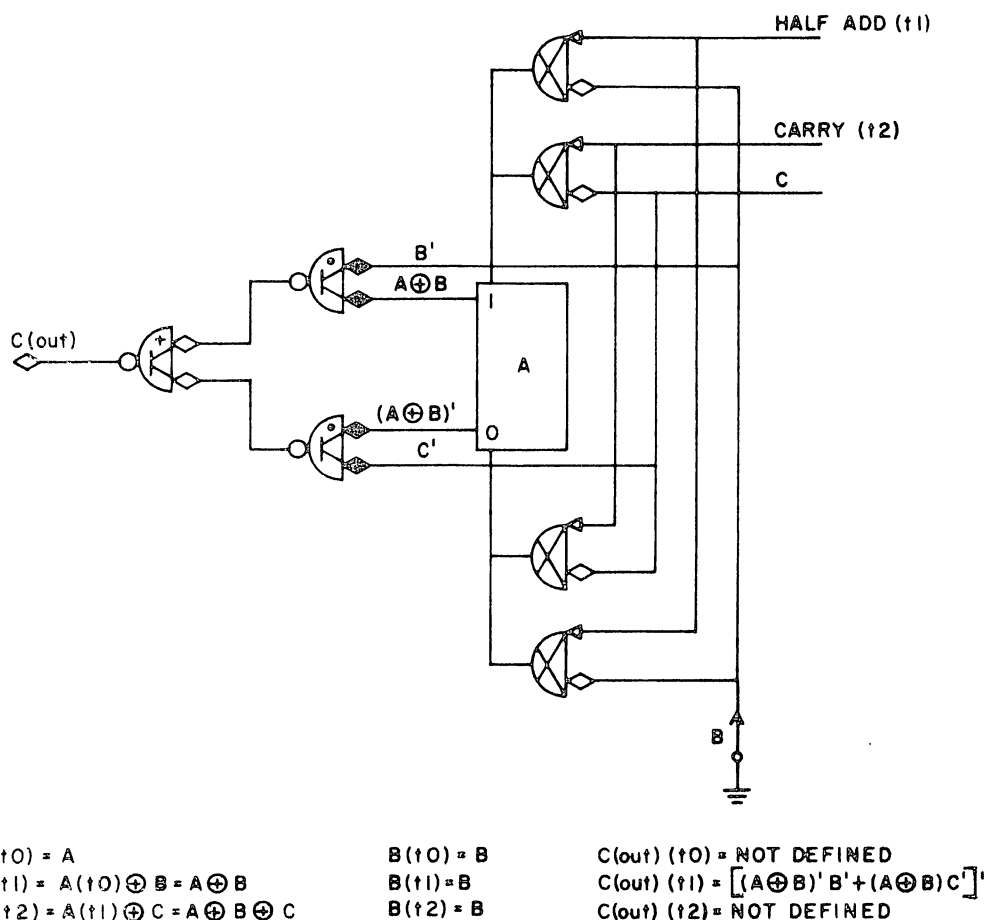


Figure 1 A Two-Step Adder Stage

Figure 2 shows a complete 4-bit adder using the toggle switches as an input. The steps involved in adding X plus Y are the following:

1. Clear the flip-flop register
2. Set X into switches
3. Half add
4. Set Y into switches
5. Half add
6. Carry

The first three steps set the flip-flop register to the number X. The next three steps add the number Y to the register.

The flip-flop register may also accumulate the sum of many numbers by repeating steps 4, 5, and 6 as often as necessary. For this reason, the flip-flop register is often termed an accumulator.

1. Construct the adder of Figure 2.
2. The table below outlines tests which can be performed on your adder to check the wiring. Calculate the anticipated results and fill in the table. Then check your adder according to the table.
3. What is the sum of the following numbers?

(a) 7 plus 3
(b) 7 plus 5

(c) 5 plus 9
(d) 8 plus 4
4. What happens if the sum is larger than 15?

	Test 1	Test 2	Test 3	Test 4
Clear				
Register holds	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
Set switches to	0 1 1 1	0 1 1 1	0 1 1 1	0 1 1 1
Half add				
Register holds	0 1 1 1	0 1 1 1		
Set switches to	1 0 0 0	0 1 0 0	0 0 1 0	0 0 0 1
Half add				
Register holds	1 1 1 1	0 0 1 1		
Carry				
Register holds	1 1 1 1	1 0 1 1		

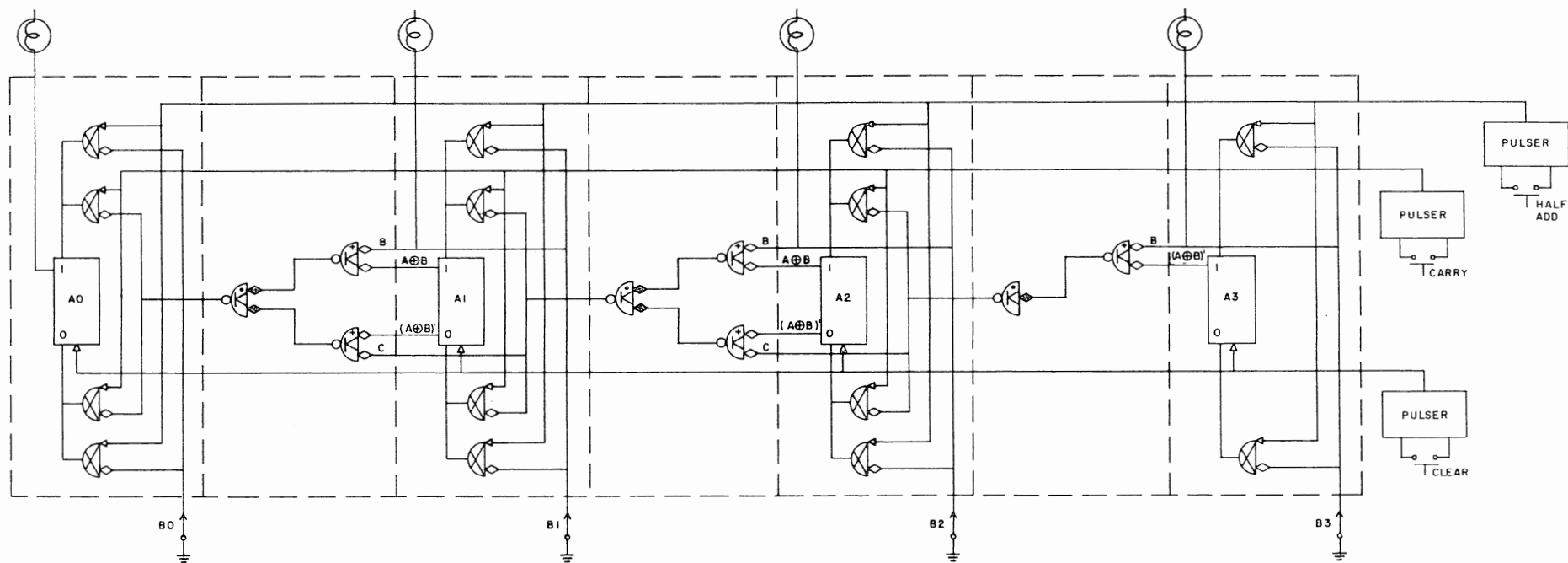


Figure 2 Two-Step Parallel Binary Adder

5. Notice that the carry signal originates at the least significant bit and propagates through a series of gates down to the most significant bit. If the delay through a gate is 50 nsec, what is the maximum time required for a carry to propagate through the 4-bit adder?

6. A signal must be present at the level input of a DCD gate for at least 400 nsec before the pulse arrives. After the gate is pulsed, it requires 80 nsec for the flip-flop output to change. What is the minimum time which can be allowed between a half add pulse and a carry pulse?

PART 2 POSITIVE AND NEGATIVE NUMBERS — TWO'S COMPLEMENT

The adder circuits in Parts 1 and 2 can be used to handle positive and negative numbers if the negative numbers are represented in 2's complement form. When the sum is positive, the output will be in the standard binary format. When the sum is negative, the output will be in 2's complement.

7. Perform the following additions by inputting the negative numbers in 2's complement form.

- | | |
|-----------------|------------------|
| (a) 3 plus (−4) | (c) −2 plus (−5) |
| (b) −5 plus 7 | (d) −6 plus 5 |

PART 3 POSITIVE AND NEGATIVE NUMBERS— ONE'S COMPLEMENTS

Since it is easier to obtain the 1's complement of a number than the 2's complement, this form is frequently preferred. However, because there are two zeros in the 1's complement representation, it is sometimes necessary to correct the results. This correction is implemented by building an end around carry gate from the most significant bit to drive the input of the least significant bit. The end around carry logic is the same as any carry logic except that it connects the two ends of the register.

8. Design an end around carry circuit for your adder and wire it in. Include a toggle switch so that you can enable or disable the end around carry.

9. Perform the following additions, representing the negative numbers in 1's complement form:

- | | |
|-----------------|------------------|
| (a) 7 plus (−5) | (c) −3 plus 3 |
| (b) −6 plus 3 | (d) −5 plus (−2) |

PART 4 SUBTRACTION

Since $X \text{ plus } (-Y)$ is the same as $X - Y$, subtraction may be performed simply by forming the complement of the subtrahend, then adding. This technique is frequently used in parallel arithmetic elements because of the large amount of circuitry that would be required to build in a separate subtractor.

10. Perform the following subtractions using 2's complement:

- (a) $5 - 2$
 (b) $5 - (-2)$
- (c) $-7 - (-3)$
 (d) $-4 - 2$

11. Perform the following subtractions using 1's complement:

- (a) $7 - 3$
 (b) $7 - (-3)$
- (c) $-7 - (-3)$
 (d) $-5 - 2$

12. If you had only a subtractor circuit, how could you perform addition?

PART 5 THE SINGLE-STEP PARALLEL ADDER

Figure 3 shows the diagram for one bit of a parallel adder which operates in a single step. This circuit is faster than the 2-step adder but requires more gate modules.

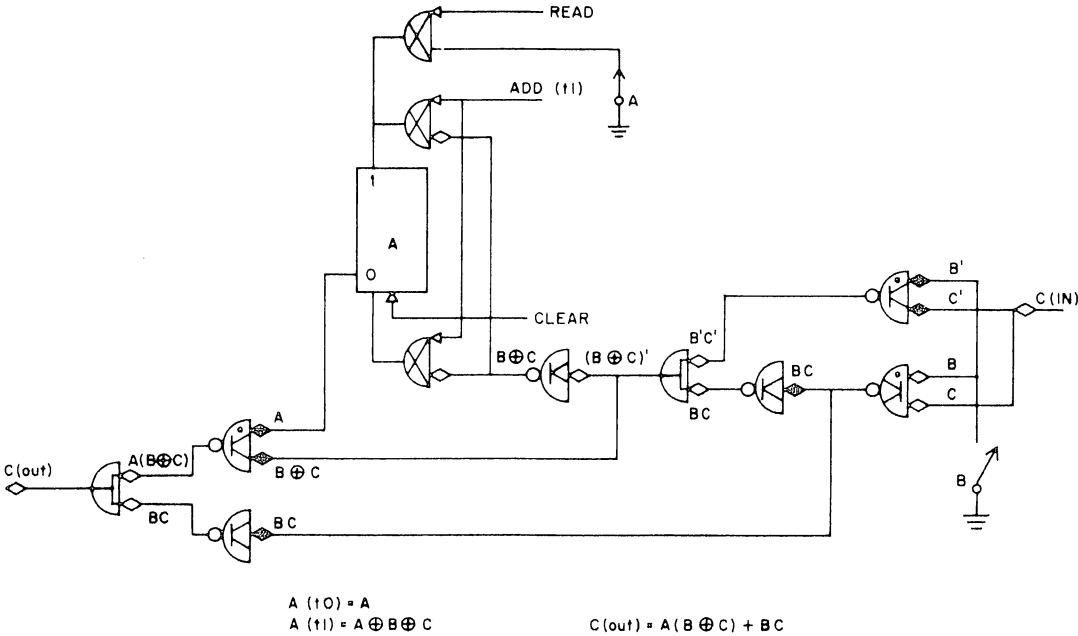


Figure 3 Single-Step Parallel Adder Stage

13. Draw a diagram for a complete 4-bit adder using the logic illustrated in Figure 3.
14. Construct an adder from your answer to problem 13.
15. Construct a test table similar to that in question 2 and check the wiring.

16. Starting with the adder equations in Experiment VI, derive the form of the equations used in the 1-step parallel adder.

17. Add the following numbers:

- | | |
|--------------|--------------|
| (a) 5 plus 3 | (c) 4 plus 9 |
| (b) 7 plus 3 | (d) 9 plus 6 |

18. Using the information given in questions 5 and 6, calculate the maximum rate at which add pulses may be applied to the adder circuit.

19. Set the toggle switches to 0011 and use the clock circuit to provide add pulses. Sketch the waveforms seen at the ONE output terminal of each of the four flip-flops. Be sure to include a time scale in your sketches.

20. With the toggle switches set to 0010 and the clock source driving the adder input, sketch the waveforms at the ONE output terminal of each of the flip-flops. Explain the difference between the waveforms found here and those found in problem 19.

EXPERIMENT IX
BINARY CODED DECIMAL ARITHMETIC

PART 1 CODE REVIEW

The digital computer, because it is an assemblage of 2-state devices, is most adept at handling the ones and zeros of the binary number system. People, on the other hand, are more accustomed to decimal numbers, and for this reason it is often desirable to build a computing system which can be operated in decimal.

To build a decimal computer with 2-state devices, it is necessary to encode each decimal digit with binary bits. Four binary bits are needed. Although only 10 of the 16 permutations possible with the 4-bit decade will be used, all are available. The number of codes that can be generated is calculated as follows:

$$16 \times 15 \times 14 \times 13 \times 12 \times 11 \times 10 \times 9 \times 8 \times 7 = \frac{16!}{6!} \simeq 2.9 \times 10^{10}$$

The choice of a code is obviously important. Desirable features of the code are: ease in performing arithmetic operation, economy of storage space, economy of gating operations, error detection and correction, and simplicity.

The 8421 code is commonly referred to simply as binary-coded decimal because the weights of the positions are the same as in the binary number system. Arithmetic operations are easily performed using the same basic method as in binary since the number sequence is the same.

In the Excess 3 code, a decimal number D is represented by the binary equivalent of the number D plus 3. The Excess 3 code is not a weighted code, but since it follows the same number sequence as binary, it is useful in arithmetic operations. Addition is facilitated since the need for a correction factor is easily detected and easily implemented. Because it is self-complementing, the Excess 3 code is also useful in subtraction.

The 2421 is a self-complementing weighted code which is commonly employed in counting systems. Other examples of 4-bit weighted codes include the 5421, the 5311, and the 74-2-1 code. All of these codes are shown in Figure 1.

Decimal	8421	Excess 3	2421	Decimal	5421	5311	7 4-2-1
0	0000	0011	0000	0	0000	0000	0000
1	0001	0100	0001	1	0001	0001	0111
2	0010	0101	0010	2	0010	0011	0110
3	0011	0110	0011	3	0011	0100	0101
4	0100	0111	0100	4	0100	0101	0100
5	0101	1000	1011	5	1000	1000	1010
6	0110	1001	1100	6	1001	1001	1001
7	0111	1010	1101	7	1010	1011	1000
8	1000	1011	1110	8	1011	1100	1111
9	1001	1100	1111	9	1100	1101	1110

Figure 1 Four-Bit Decimal Codes

PART 2 ARITHMETIC OPERATIONS WITH THE 8421 OR EXCESS 3 CODE

Because the 8421 and the Excess 3 codes follow the same number sequence as the binary number system, standard binary methods may be used. However, in binary notation 16 states are represented with 4 bits. In binary-coded decimal only ten of these states are used; therefore, special correction terms must be added to account for the six unused states.

PART 3 COUNTING

In a binary-coded decimal (BCD) counter, the corrective action is very simple. The counter is divided into 4-bit decades, and special gating is added to each decade. This gating detects the number 9 and reroutes the next count pulse so that it will reset the decade to 0 and generate a carry to the next decade.

In a down counter, the same approach is used. Starting with a standard binary down counter, the number 0 is detected, and the next count input resets the counter to the appropriate 9 designation and produces a borrow.

PART 4 ADDITION

A common method of performing BCD addition is to add two numbers in the binary adder and, if necessary, add or subtract an appropriate correction term (see Figure 2). When addition is to be performed in a decade by decade fashion (serial addition with parallel decades), either 8421 or Excess 3 code is useful. If addition is performed in parallel, however, the Excess 3 code is superior to the 8421 code.

In 8421 code the sum will be correct if it does not exceed 9. If the decimal sum is between 10 and 15, it is necessary to add 6 to the binary sum and generate a carry to the next decade. If the decimal sum exceeds 15, a carry signal is generated by the initial addition, but the correction factor 6 must still be added to the binary sum.

Addition of 8421 coded numbers has the disadvantage that a carry signal can be generated during the correction process. For this reason, each decade in the adder has to be corrected individually. Therefore it is not a desirable code in a parallel adder (see Figure 2).

When two Excess 3 numbers are added, the sum will contain an excess 6. If the decimal sum is 9 or less, it is necessary to subtract 3 in order to return to Excess 3 notation. If the decimal sum is greater than 9, the excess 6 contained in the sum cancels the effect of the six unused binary states, but it is necessary to add 3 to return to the Excess 3 notation.

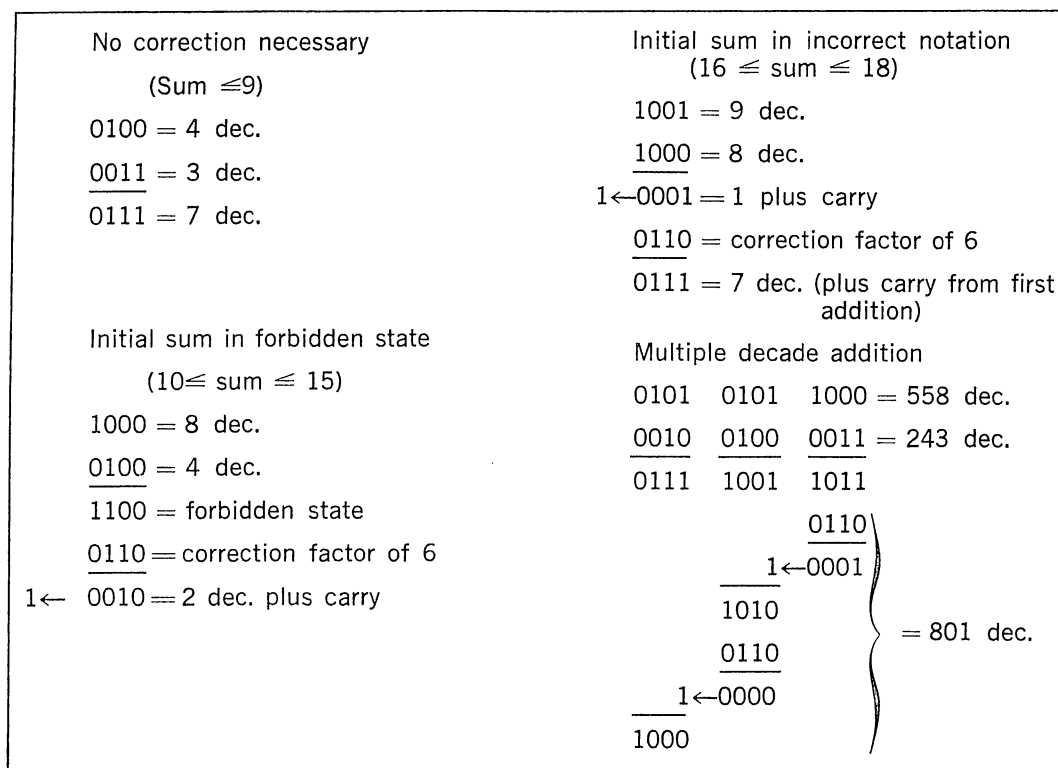


Figure 2 Addition with the 8421 Code

Whether the correction factor is 3 or -3 is determined by whether or not a carry signal appears during the initial addition. An initial carry requires a positive correction; no carry, a negative correction. The correction process will never yield an additional carry; thus simultaneous correction of all decades is possible.

The steps in performing Excess 3 addition are:

1. Add the two BCD numbers in binary fashion
2. Check each decade for a carry signal
3. Subtract 3 from each decade in which a carry has not occurred, while simultaneously adding 3 to each decade in which the carry signal has occurred.

The plus 3 correction is made by adding 0011 to the appropriate decade. Subtracting 3 from a decade is done by adding 1101 and suppressing the carry from the most significant bit of the decade. This is the method of 2's complement subtractions, described in Experiment VIII.

Figure 3 shows how the addition is performed.

Sum ≤ 9		Sum ≥ 10	
0111 = 4 dec.		1011 = 8 dec.	
0110 = 3 dec.		0111 = 4 dec.	
1101 = uncorrected sum		1←0010 = uncorrected sum	
1101 = correction factor of -3		0011 = correction factor of 3	
1010 = 7 dec.		0101 = 2 dec., plus carry from initial addition	
Multiple Decade Addition			
1000	1000	1011 =	558 dec.
0101	0111	0110 =	243 dec.
1110 ←	0000 ←	0001	
1101	0011	0011	
1011	0011	0100 =	801 dec.

Figure 3 Addition with the Excess 3 Code

Figure 4 shows how the 2-step binary adder from Experiment VIII can be converted to form a single decade of an Excess 3 code adder. An extra flip-flop is added to store the carry from the most significant bit of the decade. This is then used to determine whether the plus 3 or minus 3 correction factor is needed. The carry flip-flop gating is similar to the other stages except for a carry suppression gate which operates whenever the input from the switches is 1101, the code for the -3 correction step. Since the number 1101 is not one of the valid Excess 3 codes, this carry suppression gate will never operate during the normal addition of the two Excess 3 numbers, only during the correction process.

The steps for adding X plus Y are as follows:

1. Clear
2. Set switches to X
3. Half add
4. Set switches to Y
5. Half add
6. Carry
7. If C = 1, set switches to 0011
8. If C = 0, set switches to 1101
9. Half add
10. Carry

Steps 1-3 set the flip-flop register to the number X. Steps 4-6 add the number Y. Steps 7-10 perform the corrections so that the result is in proper Excess 3 notation.

1. Figure 5 shows a complete drawing of the 1-decade Excess 3 code adder with the carry flip-flop. Construct the circuit.

2. The table below shows a test procedure for checking the circuit wiring using the steps outlined above. The table shows the switch settings for adding 1 plus 1 and for adding 5 plus 7. Perform these steps and check that the contents of the flip-flops are correct as shown.

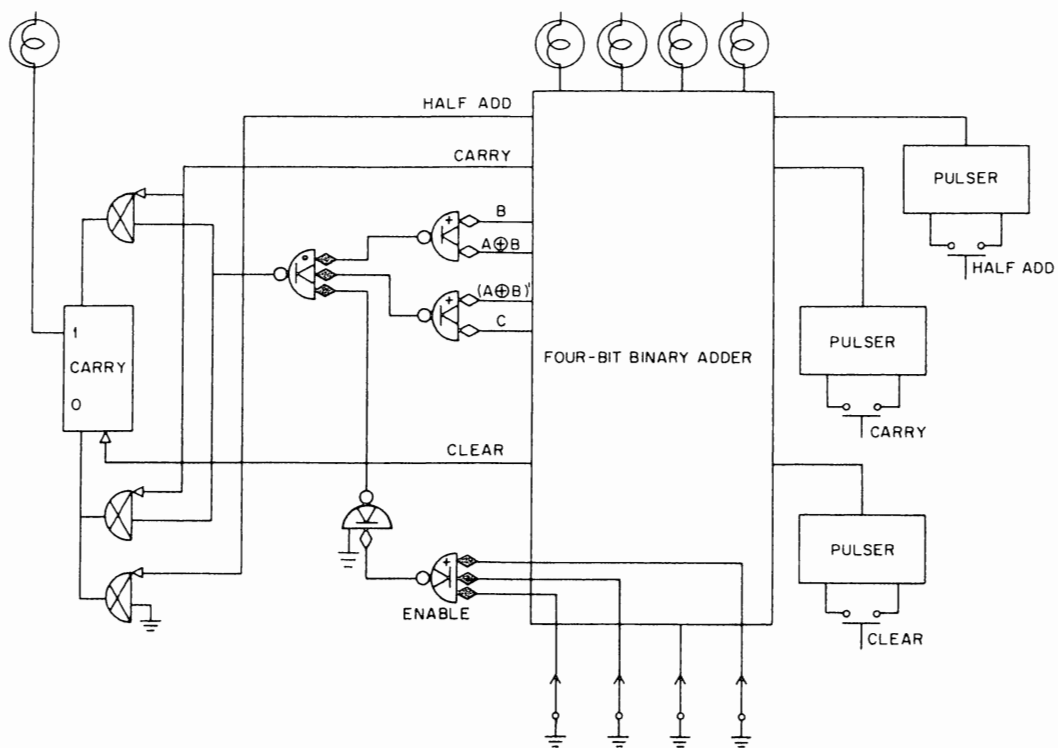


Figure 4 Excess 3 Code Adder

Step	Test 1			Test 2		
	Sw.	C	FFs	Sw.	C	FFs
1		0	0000		0	0000
2	0100			1000		
3		0	0100		0	1000
4	0100			1010		
5		0	0000		0	0010
6		0	1000		1	0010
7				0011		
8	1101					
9		0	0101		0	0001
10		0	0101		0	0101

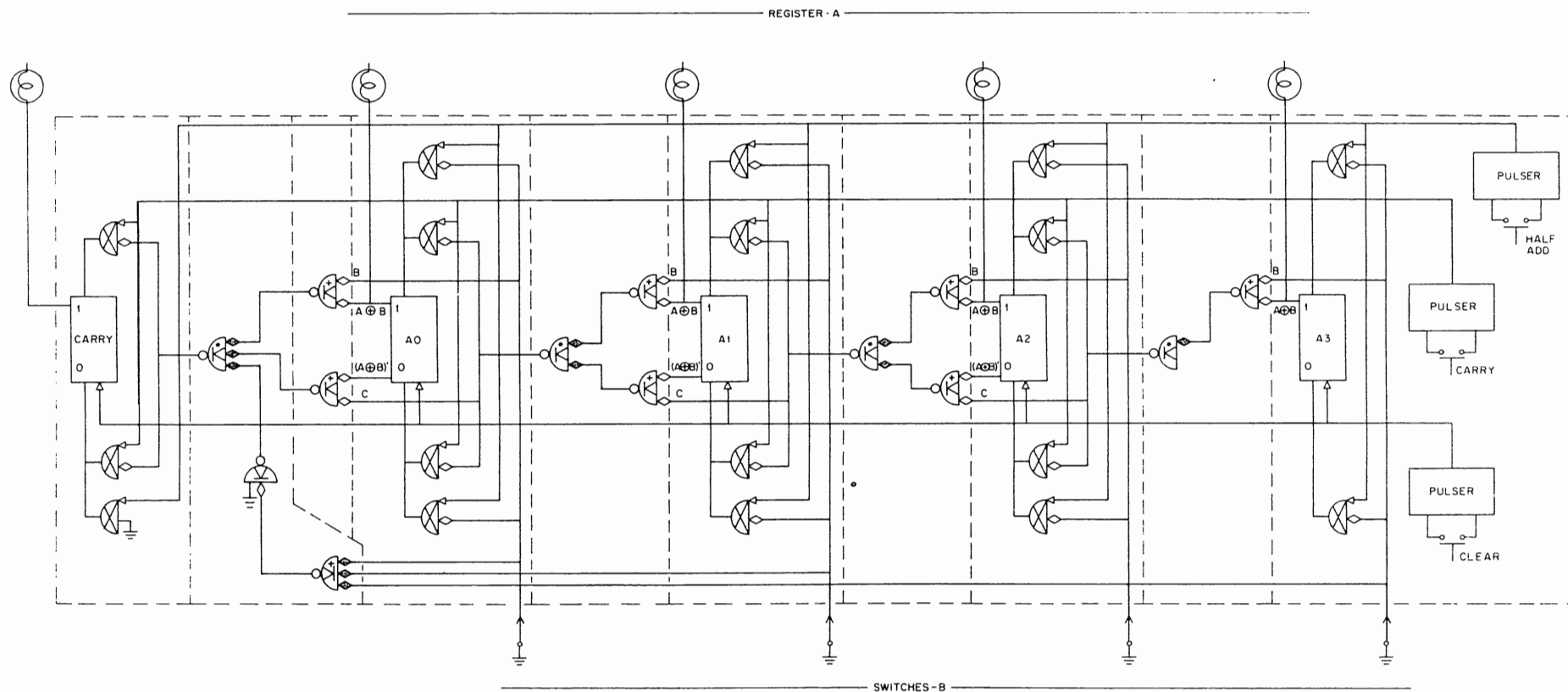


Figure 5 Excess 3 Code Adder

3. Add the following numbers and record the state of all the flip-flops at the end of steps 1, 3, 5, 6, 9, and 10. When the sum is greater than 10, only the least significant decimal digit will remain in the adder.

- | | |
|--------------|--------------|
| (a) 3 plus 2 | (c) 7 plus 6 |
| (b) 5 plus 8 | (d) 4 plus 1 |

4. Connect your adder with one of your neighbors and form a 2-decimal digit adder. Add the following numbers:

- | | |
|----------------|----------------|
| (a) 15 plus 17 | (c) 47 plus 18 |
| (b) 25 plus 43 | (d) 12 plus 29 |

PART 5 POSITIVE AND NEGATIVE NUMBERS

In Experiment VIII, you saw how negative binary numbers could be expressed in either 1's or 2's complement form. The decimal equivalent of this is the 9's or 10's complement form. The 9's complement is produced by subtracting each individual decimal digit from 9. The 10's complement is obtained by first forming the 9's complement then adding 1 to the entire number.

It is particularly easy to form a complement in Excess 3 code since the 9's complement is made simply by interchanging ones and zeros.

Figure 6 shows examples of the 9's and 10's complement. Notice that in forming the 10's complement from the 9's complement, one count is added to the entire number in Excess 3 fashion, not in binary fashion, and not to each decade.

Decimal	Excess 3	9's Comp, Dec.	9's Comp., XS 3	10's Comp, Dec.	10's Comp., XS 3
+58	0 1000 1011	-41	1 0111 0100	-42	1 0111 0101
+59	0 1000 1100	-40	1 0111 0011	-41	1 0111 0100
+60	0 1001 0011	-39	1 0110 1100	-40	1 0111 0011
+61	0 1001 0100	-38	1 0110 1011	-39	1 0110 1100

Figure 6 Examples of 9's and 10's Complement

Your adder may be used to form 9's and 10's complements if a sign bit is added as shown in Figure 7. The steps involved in forming the 9's complement of the number X are the following:

1. Clear
2. Set switches to X
3. Half add
4. Set switches to 1 1111
5. Half add

The first three steps load the number into your flip-flop register. Steps 4 and 5 complement each bit of the number. This leaves the 9's complement in the register.

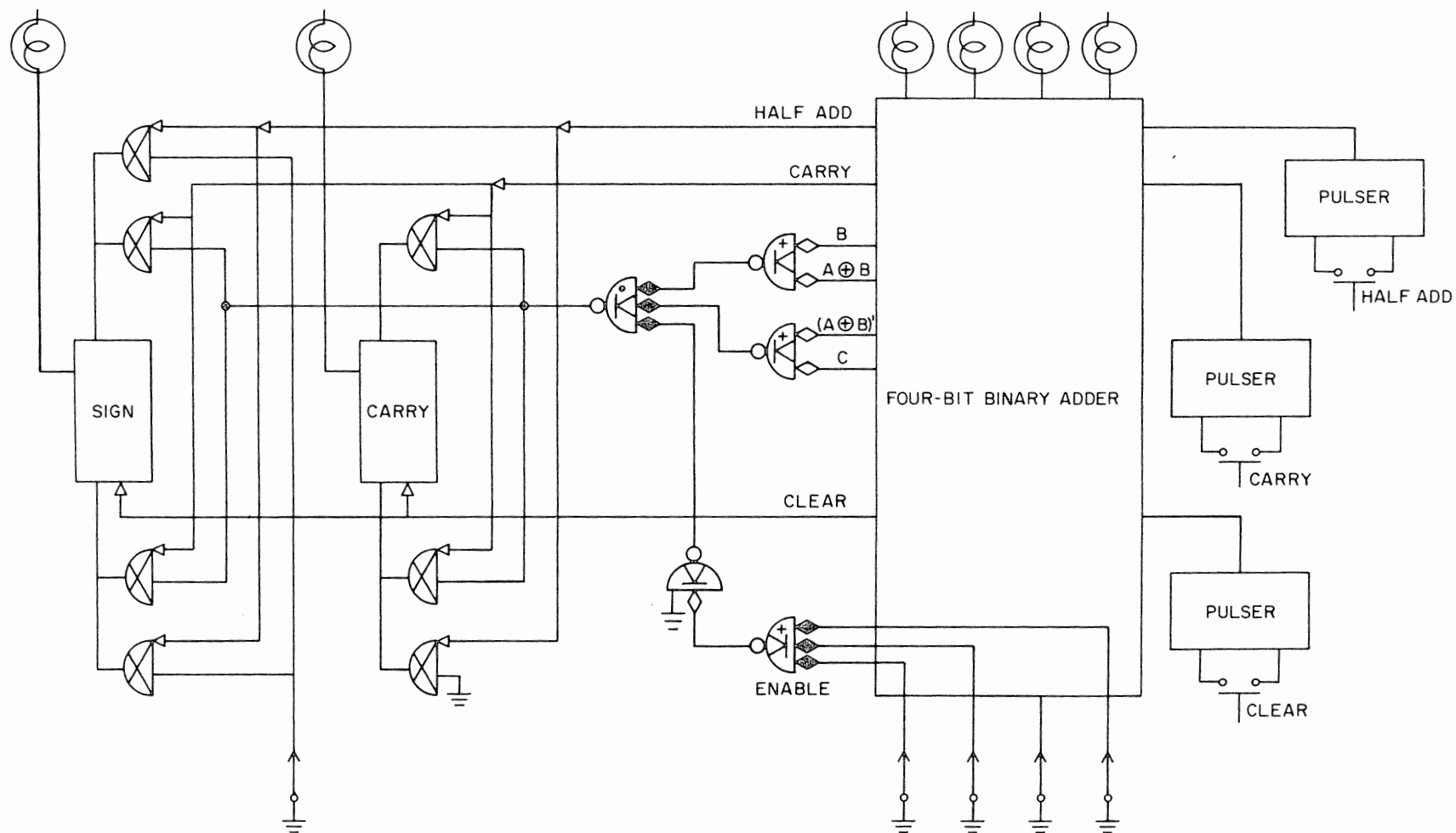


Figure 7 Excess 3 Code Adder

To then form the 10's complement, do the following additional steps:

6. Set switches to 0 0100
7. Half add
8. Carry
9. If C = 1, set switches to 0 0011
10. If C = 0, set switches to 0 1101
11. Half add
12. Carry.

Steps 6–8 are used to add 1, Excess 3 fashion, to the number, while steps 9–12 correct the addition. The final result is the 9's complement plus 1, or the 10's complement of the number X.

5. Add a sign bit to your counter.

6. Form the 9's and 10's complements of the following numbers:

- | | |
|-------|-------|
| (a) 5 | (c) 9 |
| (b) 7 | (d) 0 |

7. What would be the 9's and 10's complements of the following numbers in 8421 code?

- | | |
|-------|-------|
| (a) 5 | (c) 9 |
| (b) 7 | (d) 0 |

PART 6 ADDITION AND SUBTRACTION

As you may have realized, the decimal adder can also be used as an adder-subtractor, handling both positive and negative numbers. If the 10's complement representation is used, no additional circuitry is required. If the 9's complement representation is used for negative numbers, the end around carry gate must be attached from the sign bit to the least significant bit.

In this experiment you will use the 10's complement form. Figure 8 shows an illustration of this.

0	1000	0100	51 dec.	+51 dec.
<u>1</u>	<u>1001</u>	<u>1001</u>	(complement of 34 dec.)	<u>-34 dec.</u>
0 ←	0001	1101		
—	<u>0011</u>	<u>1101</u>	correction factor	
0	0100	1010	17 dec.	+17 dec.
<hr/>				
0	0110	0111	34 dec.	+34 dec.
<u>1</u>	<u>0111</u>	<u>1100</u>	(complement of 51 dec.)	<u>-51 dec.</u>
1	1110 ←	0011		
—	<u>1101</u>	<u>0011</u>	correction factor	
1	1011	0110	-17 dec.	-17 dec.

Figure 8 Subtraction with the Excess 3 Code, 10's Complement Notation

8. Perform the following additions and subtractions. Record your results as they are shown on the indicators and convert the results to decimal.

- | | |
|----------------------|-----------------------|
| (a) 7 plus 2 | (e) 8 minus 5 |
| (b) 9 plus (-2) | (f) 5 minus (-3) |
| (c) -7 plus 5 | (g) -5 minus 3 |
| (d) -7 plus (-2) | (h) -8 minus (-4) |

9. Perform the following additions and subtractions using the steps outlined in Part 5. Record the contents of the flip-flops at the end of steps 3, 6, and 10.

- | | |
|----------------------|-----------------------|
| (a) 5 plus 3 | (e) 9 minus 5 |
| (b) 5 plus (-3) | (f) 6 minus (-3) |
| (c) -7 plus 4 | (g) -5 minus 4 |
| (d) -6 plus (-2) | (h) -9 minus (-2) |

10. Connect your adder-subtractor with one of your neighbors to form a 2-decimal digit arithmetic element. Be sure to remove the sign flip-flop from the least significant digit. Make the following 10's complement conversions:

- | | |
|--------|--------|
| (a) 38 | (e) 42 |
| (b) 39 | (f) 13 |
| (c) 40 | (g) 98 |
| (d) 41 | (h) 0 |

(a) Perform the following additions and subtractions:

- | | |
|---------------------|----------------------|
| (a) 38 plus 37 | (e) 40 minus 40 |
| (b) 48 plus (-40) | (f) 40 minus (-39) |
| (c) -41 plus 17 | (g) -40 minus 39 |
| (d) 50 plus 0 | (h) -40 minus 0 |

(b) Perform the following additions and subtractions recording the contents of the flip-flops at the end of steps 3, 6, and 10.

- | | |
|------------------------|----------------------|
| (a) 35 plus 17 | (e) 20 minus 20 |
| (b) 40 plus (-0) | (f) 20 minus (-19) |
| (c) -45 plus 0 | (g) 30 minus 37 |
| (d) -70 plus (-17) | (h) -30 minus 0 |

PART 7 SPECIAL PROBLEMS

11. Design and construct a circuit which will perform decimal additions in 8421 code. Do not include a sign bit. Prepare a test table and check that your adder operates correctly.

(a) Prepare a write-up for your adder, including the test procedure, so that a classmate not familiar with your circuit would be able to understand its operation and check the wiring.

12. Write a report showing how addition and subtraction could be performed in 2421 code.

- (a) Design a 1-digit adder-subtractor circuit (without a sign bit) which would handle 2421 code. Prepare a test table. Construct and check your circuit.
- (b) In your report on the theory of 2421 code addition and subtraction, also include your circuit, your test table, and enough information that a classmate would be able to understand and test your circuit.

EXPERIMENT X

CODE CONVERSION

PART 1 CODE FEATURES

The choice of a good code may greatly simplify the logic circuitry. A self-complementing code allows the handling of addition and subtraction with only a few circuits. Binary coded decimal forms are particularly useful on systems which have a high rate of decimal input and output. In systems where reliability is extremely important, or where there is too large a probability of an error in transmission, error detecting and correcting codes are used. Parity bits are most common on tape units, while quite elaborate redundant codes are employed where information is to be transmitted through a noisy channel.

Decimal	Binary	Reflected Binary
0	00000	00000
1	00001	00001
2	00010	00011
3	00011	00010
4	00100	00110
5	00101	00111
6	00110	00101
7	00111	00100
8	01000	01100
9	01001	01101
10	01010	01111
11	01011	01110
12	01100	01010
13	01101	01011
14	01110	01001
15	01111	01000
16	10000	11000
17	10001	11001
18	10010	11011
19	10011	11010
20	10100	11110

Figure 1 Reflected Binary

Decimal	Reflected Decimal
0	0
1	1
2	2
3	3
.	.
.	.
.	.
8	8
9	9
10	19
11	18
12	17
.	.
.	.
.	.
18	11
19	10
20	20
21	21
.	.
.	.
.	.
99	90
100	190

Figure 2 Reflected Decimal

A special set of problems arises in systems where readout must take place on the fly. An example of this would be reading the position of a shaft angle digitizer when the shaft is in motion. Confusion could arise, for example, at the transition from the binary number 7 to the binary number 8. If the most significant bit changed slightly before the three least significant bits, the output would briefly indicate 15, a highly erroneous number. To eliminate such catastrophic errors, it is necessary to have a type of code where no more than one bit changes in going between two successive numbers. Figures 1 and 2 show reflective codes which eliminate this type of error, since only one bit (or digit) changes at a time.

In going from one piece of equipment to another, it is frequently necessary to change the code. Code changes may be done either in a parallel fashion or a serial fashion. The choice depends on whether speed or cost is more important and also on the form that the input or output data must take.

PART 2 DECIMAL CODES

The techniques of BCD conversion depend on the relation between the two codes involved. In going from 8421 to Excess 3, for example, it is only necessary to add 3. This can be done either in serial or parallel fashion.

Many times, however, the relation between the codes is not so straightforward, and it is necessary to examine the entire code group to determine what the new group should be. In this case, the code conversion of each digit is generally done in parallel. Figure 3 illustrates such a converter taking 8421 into 2421 code. The steps involved are simply:

1. Clear the converter.
2. Read in the 8421 number.
3. Pulse the convert line.

To understand how this converter operates, write out the code groups for the 8421 and 2421 codes and make a notation for each change. From Figure 4, we see that the only changes occur in the representations of the numbers 5 through 9. We also see there are only three separate types of changes that need to be made. One type of change occurs at state 5, where A replaces A', B' replaces B, and C replaces C'. At state 6 or 7, A replaces A' and C' replaces C. At state 8 or 9, B replaces B', and C replaces C'.

Decimal	8421 ABCD	2421 ABCD	Changes
0	0000	0000	
1	0001	0001	
2	0010	0010	
3	0011	0011	
4	0100	0100	
5	0101	1011	A ==> A', B' ==> B, C ==> C'
6	0110	1100	A ==> A', C' ==> C
7	0111	1101	A ==> A', C' ==> C
8	1000	1110	B ==> B', C ==> C'
9	1001	1111	B ==> B', C ==> C'

Figure 4 8421 - 2421 Codes

The next step in designing a converter is to determine the minimum identification for each state, or group of states, for which specific change must be made. In Figure 5 we see that minimum identification for the state 5 is $BC'D = 1$. States 6 and 7 exist only when $BC = 1$. States 8 and 9 exist only when $A = 1$.

State	Identification
5	BC'D
6 or 7	BC
8 or 9	A

Figure 5 Identification of States Where Changes Occur

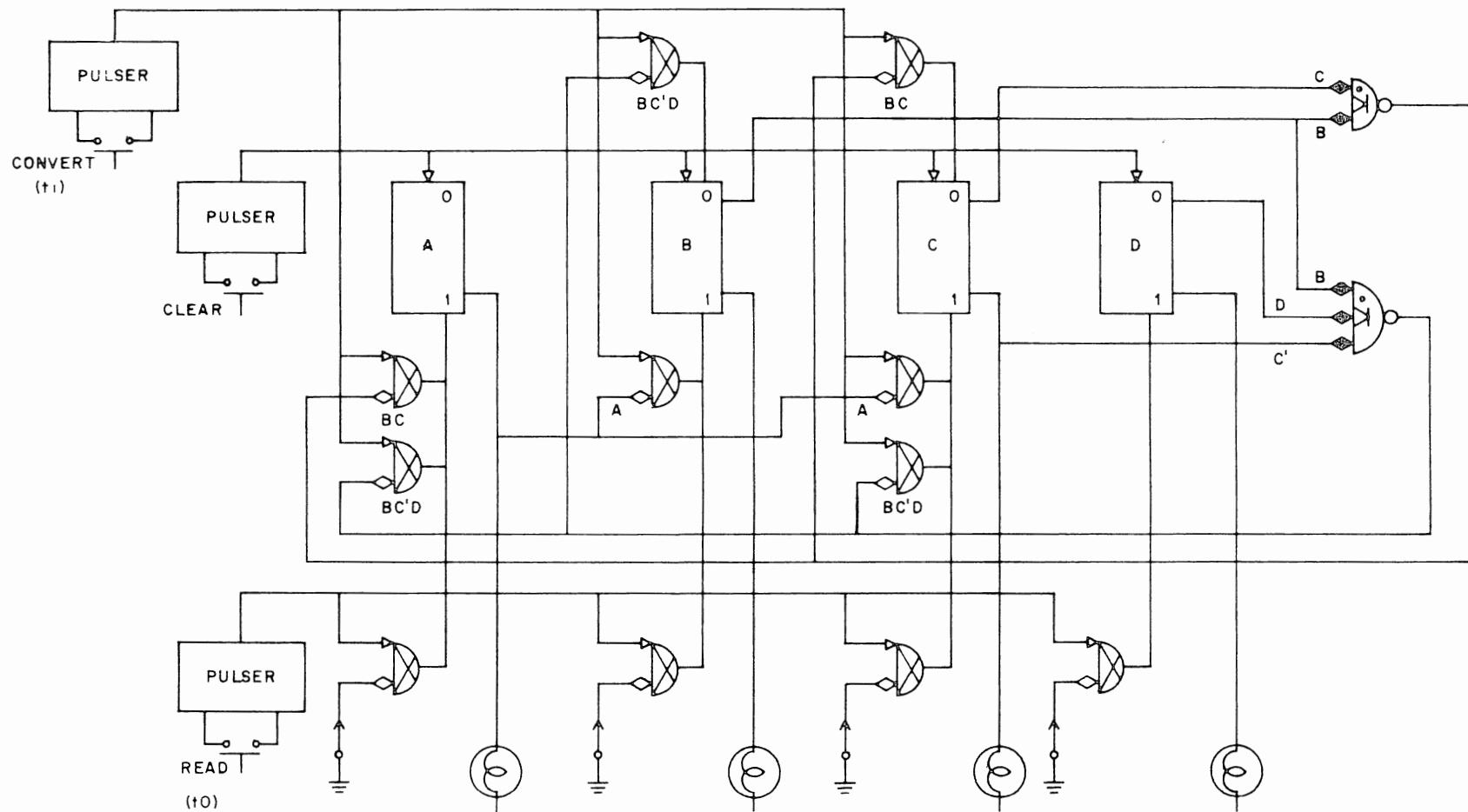


Figure 3 Code Converter 8421 to 2421

Returning to Figure 3, we can now better understand how the code converter operates. The two groups of states that are identified by more than one variable are detected by diode gates. The outputs from these diode gates and the ONE output from flip-flop A enable the gates which perform the functions outlined in Figure 4.

Notice here that the definitions of the ZERO and ONE terminals on the flip-flop have been reversed so that we can make use of all three gates on the lower side of the flip-flops. The wiring remains the same. Only the names have been changed.

The equations below Figure 3 show what is taking place in the code converter.

1. Referring to the code table in Experiment IX, design, construct, and test the following code converters:

- (a) 8421 to Excess 3
- (b) 2421 to 8421
- (c) 2421 to Excess 3
- (d) Excess 3 to 8421
- (e) Excess 3 to 2421

2. Calculate the maximum load applied to any of your flip-flops in your code converter from question 1. Compare this with the rated driving ability.

PART 3 ERROR DETECTING CODES

In systems where a single error might be expected to occur, a parity bit is often used. This is an extra bit added to the word so that the total number of ONES will be always odd or always even.

Figure 6 shows a circuit which will generate even parity. If there are an even number of ONES in the four information bits, the parity bit will be ZERO. But, if there are an odd number of ONES in the four information bits, the parity bit will also be ONE so that the total will be even.

The circuit of Figure 6 operates in parallel using exclusive OR circuits. Each pair of bits is exclusively ORed, then the outputs of the exclusive OR circuits are further exclusively ORed, etc.

Parity may also be generated quite easily for serial information. A special parity flip-flop is complemented each time a ONE is received. When the entire word is circulated, the parity flip-flop will contain the parity bit, either odd or even, depending upon whether it was initially cleared or set.

3. Construct and test the circuit of Figure 6. Without adding any more gates, change the circuit from even parity to odd parity.

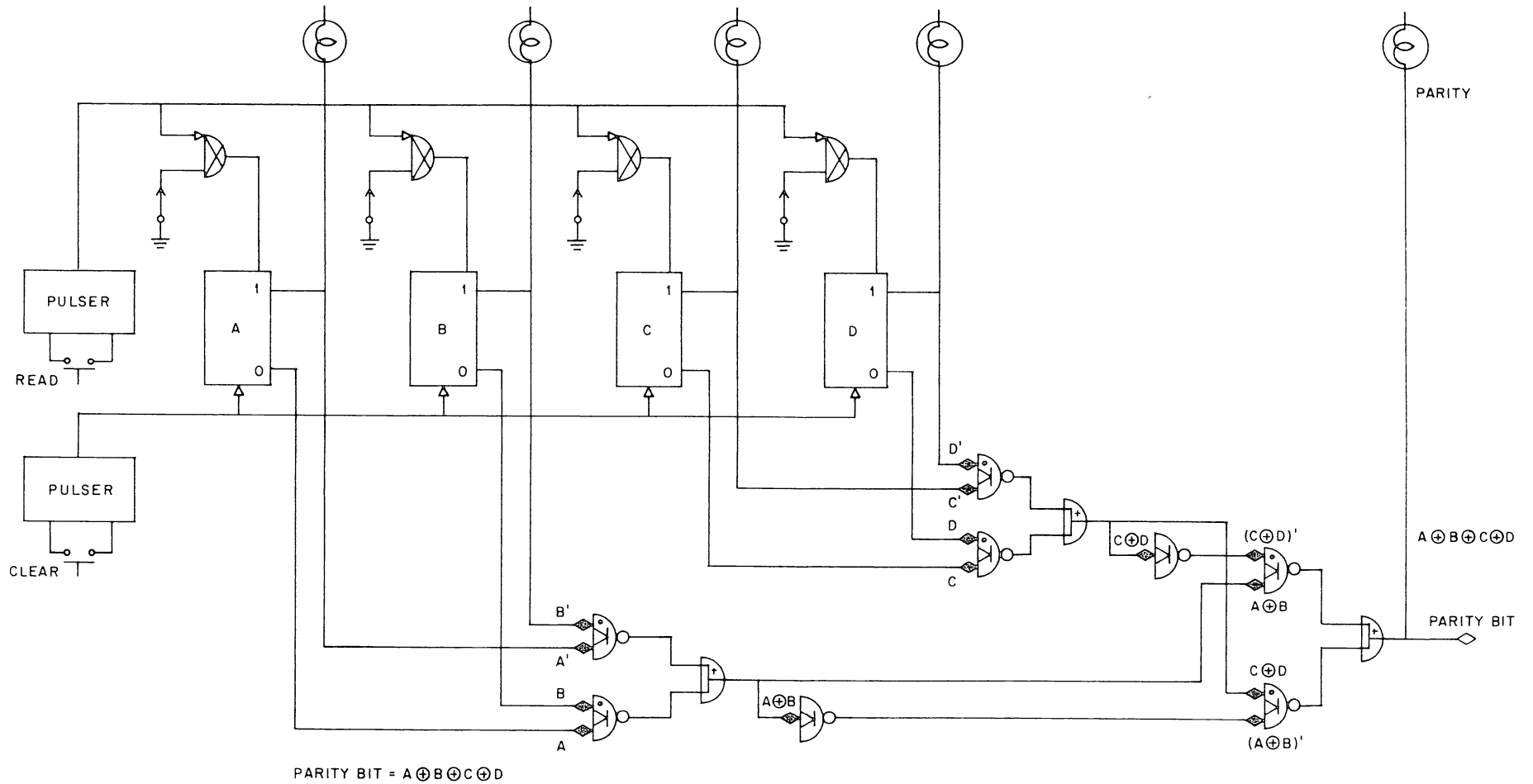


Figure 6 Parity Generator

4. Extend the circuit of Figure 6 to handle six information bits, generating odd parity. Construct and test the circuit.
5. Construct a serial parity generator for odd parity.
6. The 2 out of 5 code is a binary-coded decimal form which also allows the detection of a single error. This code is shown in Figure 7. Design a serial circuit which detects a single error in this code.

Decimal	2 out of 5
0	00011
1	00101
2	00110
3	01001
4	01010
5	01100
6	10001
7	10010
8	10100
9	11000

Figure 7 2 out of 5 Code

PART 4 REFLECTIVE CODES

Reflective codes avoid errors when reading a changing number, such as the output of the shaft angle digitizer described in Part 1. Because these codes are difficult to use in arithmetic operations, they are normally converted to standard binary. The reflected binary code, illustrated in Figure 2, is most commonly used. The rules for converting from this code to binary are:

1. The most significant digit is identical in reflected and standard binary.
2. If a digit is a ONE after being converted to standard binary the following (or less significant) digit is complemented.

Figure 8 shows a reflected-to-standard binary code converter that operates in parallel. This circuit is similar to a counter in that information must propagate down a chain of bits. In this case, propagation starts at the most significant bit and propagates to the least significant bit, providing a complement signal whenever the exclusive OR of all bits thus far is a ONE.

7. Construct and test the reflected-to-standard binary code converter of Figure 8. Using the information from Experiment VIII questions 5 and 6, calculate the minimum time which must be allowed between a read signal and a convert signal.
8. Reflected binary-to-standard binary code conversion lends itself very well to a serial technique because the state of each bit depends only on the more significant bits. Design, construct, and test a serial reflected-to-standard binary code converter.
9. To understand why reflected binary is not normally used in arithmetic operations, design a 3-bit counter which operates in reflected binary.
 - (a) Try to minimize the cost of your reflected binary counter. Assume that a flip-flop costs twice as much as a diode gate and a diode gate costs twice as much as a DCD gate.

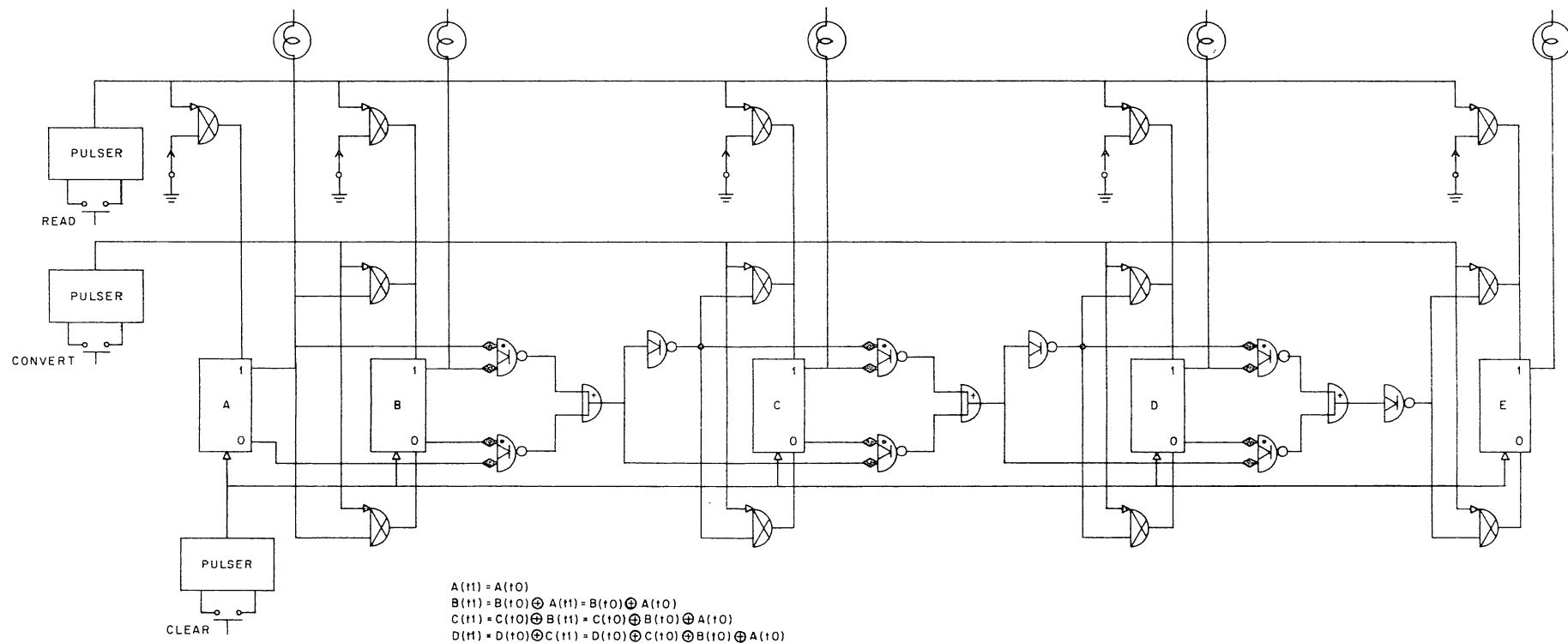


Figure 8 Gray Reflected Binary to Binary Code Converter

PART 5 SPECIAL PROBLEM

10. Design a reflective BCD system. Design a circuit which will convert this code to Excess 3 notation. Construct and test your circuit. Prepare a report including a description of the code, the conversion circuit, the operation of the conversion circuit, and a test procedure.

EXPERIMENT XI

CONTROL

PART 1 THE USE OF CONTROL CIRCUITRY

In the experiments so far whenever a series of different operations were to be performed, you controlled the operation by depressing push-buttons in a fixed sequence. For example, with the two-step parallel adder, you depressed buttons which performed the functions clear, half-add X, half-add Y and carry. In a computer or other high-speed digital systems, the same sequence of events must take place, but at a much higher rate. Thus, the computer must contain a control section which generates a series of pulses in the right sequence, and at the right times, just as you did with the buttons.

The heart of the computer control system is a pattern generator which produces a fixed pattern of pulses on a series of lines. Each line corresponds to an operation, such as the clear, half-add and carry operations.

Associated with the pattern generator is gating circuitry which enables the pulses whenever the desired operation is being performed, or inhibits them when the operation is not desired. For example, a computer instruction to add would enable one set of pulses while a computer instruction to multiply would enable a different set.

PART 2 GENERATING THE PATTERN

Pattern Generators can be made in a variety of different ways. Many of these you have already studied. The ring counter, of Experiment IV, Part 2, will produce a series of N discrete states, then automatically restart itself and produce the same series of N states again. The ring counter requires a separate flip-flop for each state it generates, which for a complex pattern, can mean that many flip-flops are used. However, it is quite simple to interpret the output of a ring counter. To determine if a given state is present, it is only necessary to see whether or not the corresponding flip-flop holds a ONE. This is shown in Figure 1a.

The switched tail ring counter, of Experiment IV, Part 2, may be used to generate any pattern which has an even number of states. It is more economical than the ring counter because it requires only half as many flip-flops. But in order to determine whether or not a given state is present, it is necessary to look at two flip-flops as shown in Figure 1b.

Counter circuits (Experiment I) are also used as pattern generators. The counter configuration has the advantage only N flip-flops are required to generate 2^N states. However, to determine if a given state is present, it is necessary to check the state of all flip-flops, which requires a large amount of decoding circuitry (see Figure 1c).

(a)		(b)		(c)	
4-Bit Ring Counter		4-Bit Switched Tail Ring Counter		4-Bit Counter	
States ABCD	Identification	States ABCD	Identification	States ABCD	Identification
1000	A	0000	D'A'	0000	A'B'C'D'
0100	B	1000	A B'	0001	A'B'C'D
0010	C	1100	B C'	0010	A'B'C'D'
0001	D	1110	C D'	0011	A'B'C D
		1111	D A	0100	A'B C'D'
		0111	A'B	0101	A'B C'D
		0011	B'C	0110	A'BC D'
		0001	C'D	0111	A'BC D
				1000	AB'C'D'
				1001	AB'C'D
				1010	AB'C D'
				1011	AB'C D
				1100	ABC'D'
				1101	ABC'D
				1110	ABCD'
				1111	ABCD

Figure 1 Pattern Generators

The standard counter has one main disadvantage. The number of states must be an integral power of two. For this reason, special counters are often used. These operate in the standard binary fashion but automatically reset themselves after a fixed number of states has been generated. For example, Figure 2 illustrates a count-of-5 circuit.

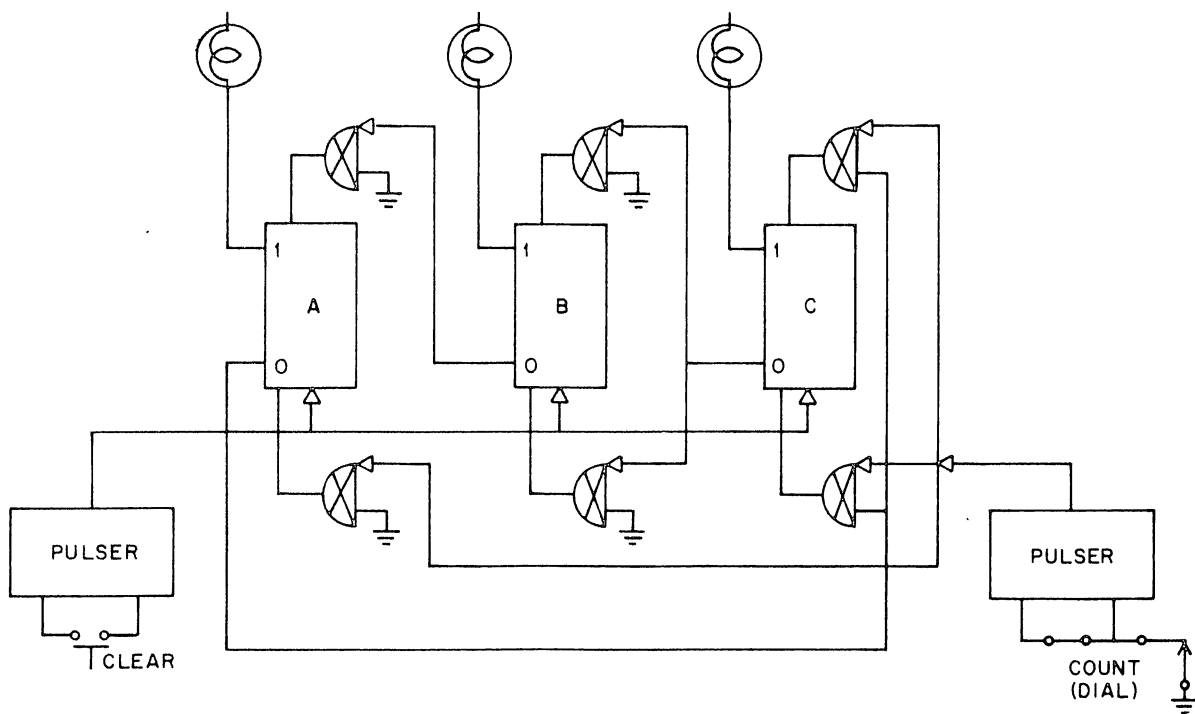


Figure 2 Count-of-5 Circuit

This same counter, with an additional flip-flop added, was used in Experiment II to make a binary coded decimal counter in the 8421 code. Moreover, this circuit may be extended to generate any pattern of $2^{M(2^{N+1} + 1)}$ states by adding flip-flops either at the beginning or at the center of the circuit as shown in Figure 3.

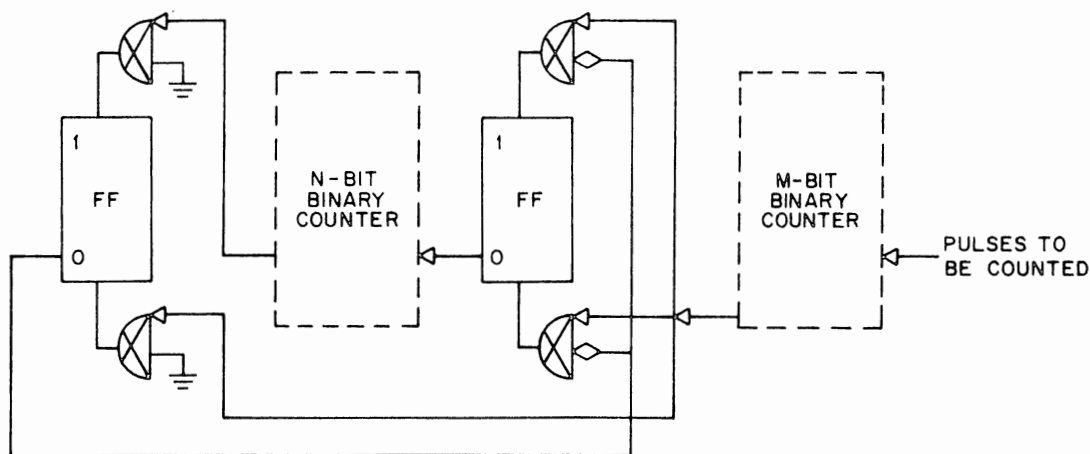


Figure 3 Count-of- 2^M ($2^{(N+1)} + 1$)

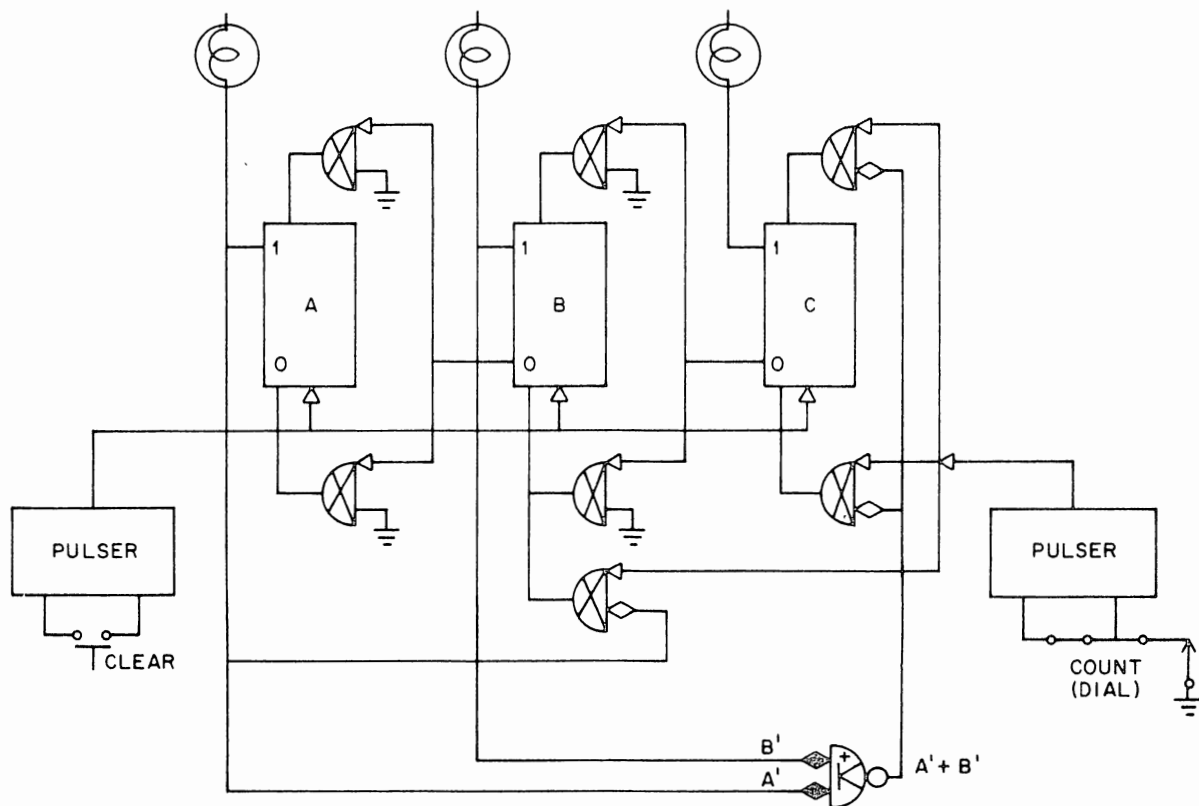


Figure 4 Count-of-7

Although not all special counts are as easy to obtain as the count-of-5, count-of-10 family, any number may be obtained if extra gates are added to detect the final state and produce all the necessary enable and inhibit signals. For example, Figure 4 shows a count-of-7 circuit. The operation can be seen by assuming the circuit starts at 0. It will count in standard binary fashion until the number 110 is reached. This will be detected by the diode gate which will inhibit flip-flop C. At the same time, the output of flip-flop A and the internal conditioning on flip-flop B will enable the lower clear input on B. Thus, the next pulse will return the counter to the number 000.

1. Construct the count-of-5 circuit shown in Figure 2. Driving the input from the dial, test that your circuit operates correctly.

(a) What would happen if your counter came to state 5 when the power was first turned on and if the count input was operated without first clearing the counter? Predict the result then test it by connecting the clear input so that it resets the counter to 5.

(b) Repeat (a) for state 6.

(c) Repeat (a) for state 7.

(d) Drive the input from a clock and observe and sketch the waveforms at the ONE output terminals of flip-flops A, B and C. Can you tell that each of these patterns is repetitive and is made up of five segments? Why or why not?

PART 3 THE PULSE AMPLIFIER

Once the pattern is generated, the pulses must be gated out onto the appropriate lines to perform the various functions. A special circuit, called a pulse amplifier or PA, is used to do this gating.

Figure 5 shows the diagram for the type R602 pulse amplifier used in the Logic Laboratory. The inputs to this circuit are very similar to those on the flip-flop. There are two diode-capacitor-diode gates, each of which have a pulse and a level input. There is also a direct input which can receive a pulse from a clock, from another pulse amplifier, from the push-button pulsers or from a gate. Whenever the input conditions are met (that is, there is a pulse at the direct input or there are both a pulse and a ground level on a DCD gate) the pulse amplifier circuit will produce a standard pulse at its output. The output pulse will go from a base of -3 volts to ground, remain at ground for 100 nanoseconds and then return to -3 volts.

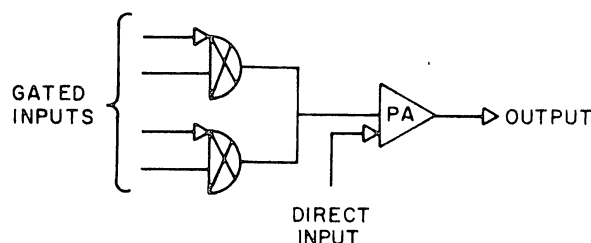


Figure 5 Pulse Amplifier Diagram

2. To see the operation of the pulse amplifier, connect its direct input to a clock, and observe the output on the oscilloscope.

Although the pulse amplifier is used for gating, it has several unique advantages that distinguish it from the simpler diode gate. It standardizes pulses in duration as well as amplitude. Any input signal between 40 and 100 nanoseconds will be stretched to 100 nanoseconds and any input signal of more than 100 nanoseconds will be reduced to 100 nanoseconds.

The output of the pulse amplifier (see appendix A) is capable of driving up to 70 MA of external load. Thus the circuit is particularly useful in large computers where the registers contain many flip-flops and the load that must be driven is quite heavy.

Because the inputs to the pulse amplifier circuits are DCD gates which have a delayed input and a differentiating input, similar to those used on flip-flops, the pulse amplifier is frequently used to expand the number of inputs to a flip-flop. A PA may drive a flip-flop either through its output terminal (as in Figure 6) or through its input.

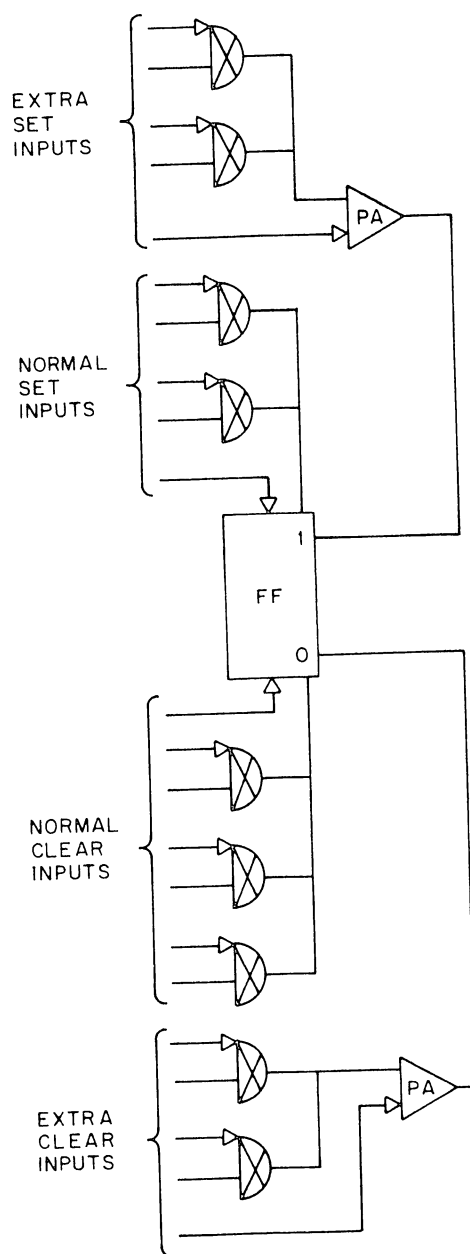


Figure 6 Pulse Amplifier Driving a Flip-Flop Through Its Outputs

The diode-capacitor-diode gates on the pulse amplifier have two inputs. The output occurs only when the conditions of both inputs are met. If the level input is driven from a complex gate network, there will be a large number of conditions which are required for a pulse output. All of these conditions are represented in the pulse output. Thus the pulse amplifier can be used to generate a powerful signal which carries a considerable amount of information to a variety of places.

The output circuit of the R602 is a transistor with a clamped load resistor similar to the output circuit of a gate. If two or more PA outputs are wired together, this will form a pulse OR gate as shown in Figure 7. Unlike wired OR gates for logic levels, however, this cannot be reversed to form a wired AND, since "no pulse AND no pulse" will not initiate any action.

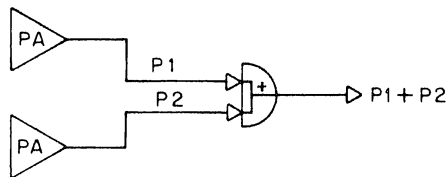


Figure 7 ORing the Outputs of Pulse Amplifiers

Figure 8 illustrates the use of the pulse amplifier in a pulse distributor circuit with two output lines. Line 1 produces outputs at times 0, 1 and 3, while Line 2 produces pulses at times 2 and 4. The toggle switches allow you to select whether or not the corresponding outputs pulses are to occur.

3. Add the pulse amplifier and gating circuitry to your count-of-5 circuit as shown in the Figure 8. Driving the counter input from the clock circuit, observe the output of each of the pulse amplifiers on the oscilloscope and sketch the waveforms. When all the switches are closed, can you tell from looking at each pulse amplifier output that it is part of a repetitive pattern which contains 5 states?

PART 4 THE DELAY (ONE-SHOT)

The delay one-shot is similar to a flip-flop circuit, but it has only one stable state. It always tries to remain in this state. If you change the state of the one-shot, it will return automatically to its original state after a short period of time.

Figure 9 shows how a delay one-shot is constructed. Notice that it is identical to a flip-flop except that one of the feedback lines is AC coupled, so that the signal will decay after a short time. The normal, or stable, state of the circuit is with points A and B at -3 volts and point C is at ground. If A is suddenly grounded, the other side of the capacitor (B) will also go to ground and the output of that gate (C) will go to -3 volts. This will hold point A at ground. However, as current starts to flow through the capacitor, point B will begin to approach -3 volts. When it becomes sufficiently negative to switch the lower gate, all the terminals will change and the one-shot will go back to its normal state.

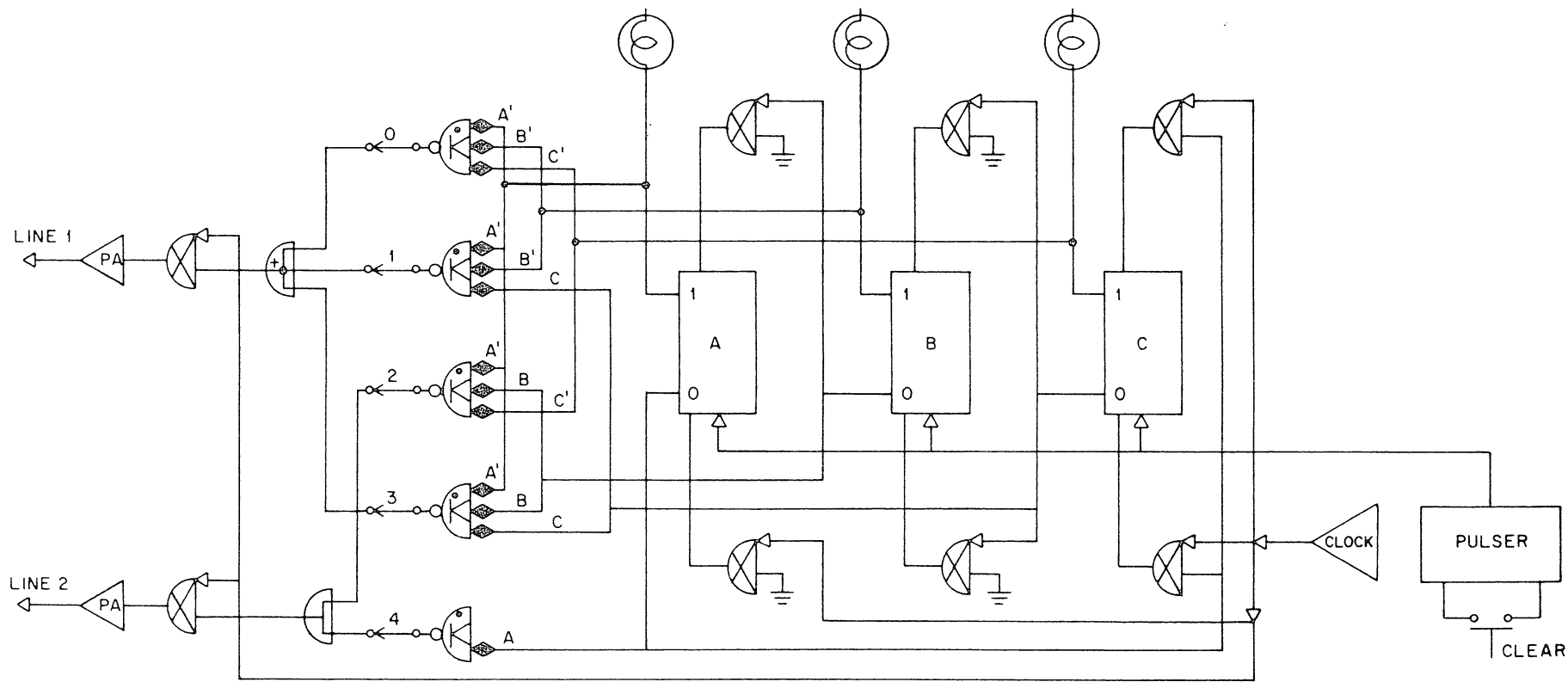


Figure 8 5-State Pulse Distributor

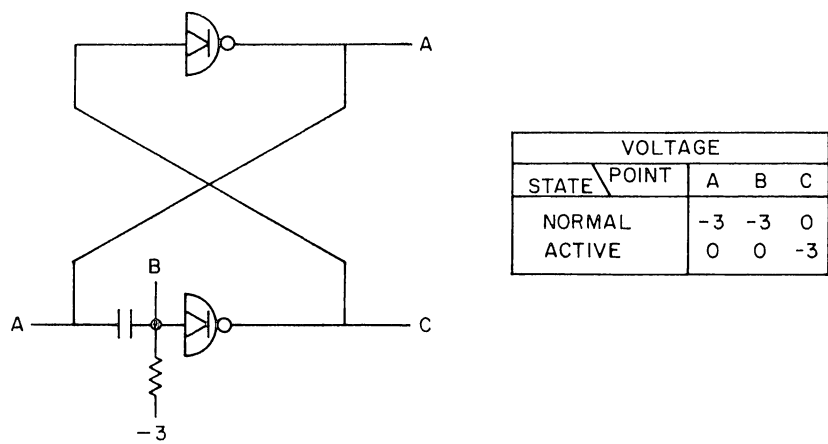


Figure 9 Delay (One-Shot)

The time required for the delay unit to return to its normal state depends upon the values of the capacitor and the resistor in the feedback network. By changing these values, the charging or delay time, can be adjusted to any desired value.

Figure 10 shows the symbol for the delay one-shot, Type R302, which is included in the Logic Laboratory. This circuit has a capacitor of 220 pf (picofarads, sometimes called micromicrofarads) and an internal resistor of 1000 ohms. The resistor and the capacitor are not connected. To complete the circuit, they should be jumpered together using either an internal potentiometer (as shown with the dotted line) or using one of the potentiometers from the indicator switch panel (as shown in Figure 10b). In this way, the potentiometer may be used to change the duration of the delay. For more extreme changes in the delay, external capacitors may be attached in parallel with the internal capacitor.

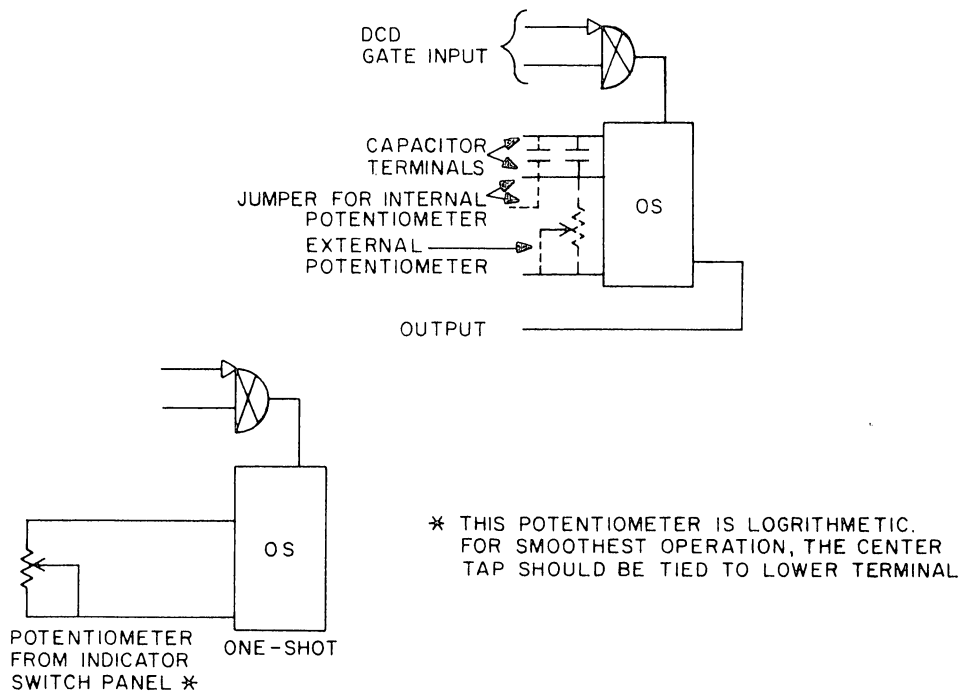


Figure 10 Using the Delay One-Shot

The output of the R302 will be at ground when the delay is in its normal state and at -3 volts when the delay is active. The input gate is a DCD circuit similar to those on the flip-flop.

4. To see the operation of the delay, connect it as shown in Figure 10b and drive the input from the clock. What are the minimum and maximum values of the delay which you get by changing the potentiometer?

PART 5 A PULSE DISTRIBUTOR MADE WITH DELAYS

Delay one-shots are used to make pulse distributors by connecting them in a circle as shown in Figure 11. When the power is first turned on, all of the delays will be in their normal states. If a push-button is used to turn on D1, its output will go negative for the duration of the delay. Then it will go positive, automatically triggering D2; which will go negative for the duration of its delay, then automatically trigger D1 as it returns to ground. This continues around the chain with each delay driving the next indefinitely. One of the advantages of the delay system is that the delays can be set for an arbitrary duration. Thus, the pulses sent out by the pulse distributor can be at any time, not just at multiples of a fixed frequency.

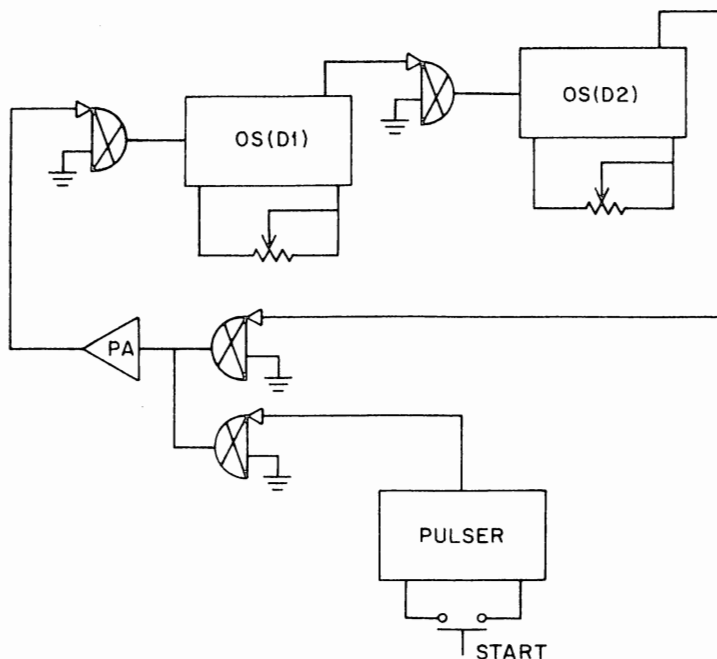


Figure 11 Pulse Distributor

5. Construct a simple distributor circuit as illustrated in Figure 11. Set D1 for one microsecond. Observe and sketch the output waveforms at D1 and D2 with D2 set at
 (a) its minimum.
 (b) its maximum.

6. By attaching external capacitors, the delay range may be increased. Add a 220 pf external capacitor on to the terminals of D2. What are the new minimum and maximum values of the delay range?

PART 6 SPECIAL PROBLEMS

7. Using the techniques described in Figure 3 design, construct and test a count-of-9 circuit.

- (a) Drive the counter from a clock and observe and sketch the waveform at the ONE output terminal for each of the flip-flops.
- (b) Add two pulse amplifiers so that you gate pulses out of PA1 at times 0, 1, 3 and 7, and out of PA2 at times 4 and 8. From the pulse amplifier output waveforms can you tell that this is a 9-state repetitive pattern?

8. Design, construct and test a count-of-11 circuit.

- (a) Try to minimize the cost of your circuit assuming that each gate circuit sells for D dollars while the flip-flop circuit, including all five of its DCD gates, sells for 3D dollars. What is the cost of flip-flop and gate circuits in your counter?
- (b) Drive the input of the counter from the clock. Observe and sketch the output waveforms from each of the flip-flops. Can you tell that each waveform is generated by an 11-state pattern generator?
- (c) Add two pulse amplifiers to your circuit. Distribute pulses from one pulse amplifier when the counter is in state 0, 3, 7, 9 or 11. Distribute pulses from the other PA when the circuit is in state 2 or 6.
- (d) Using two pulse amplifiers to distribute pulses, gate the outputs on PA1 for all even numbered counts; on PA2 for all odd numbered counts.

9. Design a count-of-13 circuit. Sketch how the waveform should appear at each flip-flop ONE terminal. Instead of connecting a push-button to your counter input, connect the clock circuit and test your counter by directly observing the flip-flop output waveforms on the oscilloscope.

10. The delay time of the one-shot is proportional to the total capacitance multiplied by the total resistance in the RC network. Try adding external capacitors (between 220 and 22,000 pf) and external resistors (between 0 and 20,000 ohms) and measure the delay times. Plot the delay times versus RC and determine the constant of proportionality. (Remember that C is equal to your external value plus the internal value of 220 pf, and that R is equal to your external value plus the internal value of 1000 ohms.)

11. In Experiment VIII you constructed a two-step parallel adder using the logic shown in Figure 2 of that experiment. In order to operate the counter there were 6 steps involved. These were:

1. Clear the flip-flop register
2. Set X into the switches
3. Half-add
4. Set Y into the switches
5. Half-add
6. Carry

Modify the adder logic so that it receives the number X from four of the toggle switches and the number Y from the other four toggle switches or your Logic Lab panel. Using delay one-shots, design and construct a pulse distributor to perform the entire operation automatically after you set both numbers into the switch registers and depress one pushbutton. Construct and test your adder circuit; then write a short report (one or two pages) describing how your circuit operates and how you tested it.

EXPERIMENT XII

TIMING

PART 1 SIMULTANEOUS SIGNALS

Timing considerations are extremely important in designing digital logic. Although the digital circuits are very fast, each operation still requires a finite amount of time to be completed, and once completed, the capacitors and inductors in the circuit must be restored to their original voltage or current before another action can safely be initiated.

If a digital circuit receives an input asking it to perform a new action before the previous action has been completed, or if the capacitors and inductors have not had time to restore their proper levels, then the results will be unpredictable. The action may or may not take place depending on the amplitude of the input signal, the tolerances of the components in the circuit, the quality of the AC voltage supply, the temperature, the humidity, the age of the circuit and so forth.

Timing is particularly important when the signal in question goes to many digital circuits. Some circuits may respond while others may not. Imagine, for example, that the number 0111 is to be changed to 1000 and that all of the flip-flops respond correctly except the one which contains the most significant bit. The computer will then end up holding the number 0000.

To avoid timing problems, the logic designer must know when each of his signals will occur, and if he finds two or more that will occur simultaneously, then he must make provisions to avoid catastrophic errors. He may do this by delaying the offending signal, by inhibiting it or by overriding it. He must do this clearly and without an ambiguity. If the suppression of the offending signal is not handled cleanly, the result could be a split or partial pulse which would be sufficient to trigger some circuits but not others. This would be equally catastrophic.

Timing problems are possible even in the simplest circuits. Figure 1 shows a 3-bit counter with a clear input. Assume that the counter is in the state 001 when a clear signal is given. As flip-flop C goes from the ONE to the ZERO state, its ZERO output terminal will go to ground, generating a carry to flip-flop B. Since flip-flop B is already in the ZERO state, the carry will attempt to set it at the same time as the clear pulse is attempting to clear it. Thus flip-flop B may go either to the ONE or the ZERO state.

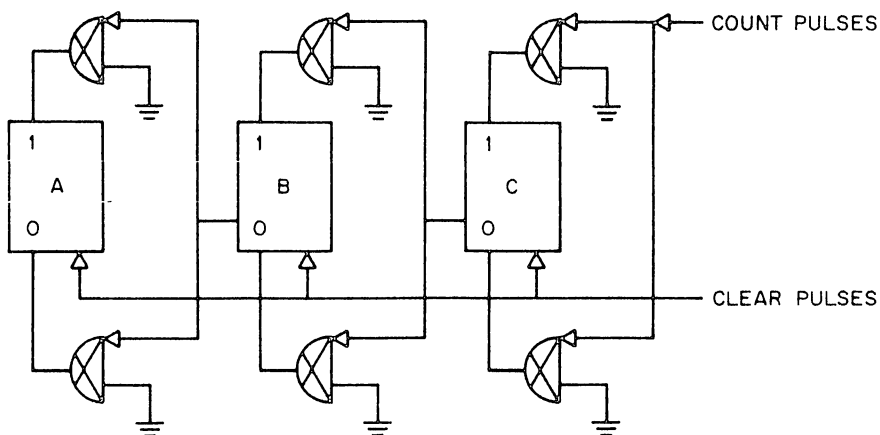


Figure 1 Simultaneous Signals

In the counter circuits that you constructed thus far, these difficulties have been avoided by using a push-button generated clear signal that is of long enough duration to override the carries. The push-button, even when activated as rapidly as possible, produces a signal of several milli-seconds. Since the signal comes into a direct input terminal on the flip-flop, it will continue trying to clear the flip-flop as long as it is present. Because the time required to generate a carry is only about 70 nanoseconds, the clear pulse will be present long after all the carries in the chain have died away, and hence will override the carries.

1. To see what would happen if a clear signal were not long enough to override all carries, connect the circuit as shown in the Figure 2. In this circuit, the clear and count signals are still generated by pulsers; however, they both go to DCD gates which differentiate the leading (positive-going) edge of the signal to produce a pulse of about 70 nanoseconds duration. The clear signal goes through a pulse amplifier which assures that it is full amplitude, and also stretches it slightly to 100 nanoseconds. Try setting the counter to different numbers between 1 and 7, then clearing it. Note what happens when you attempt to clear out each of these numbers.

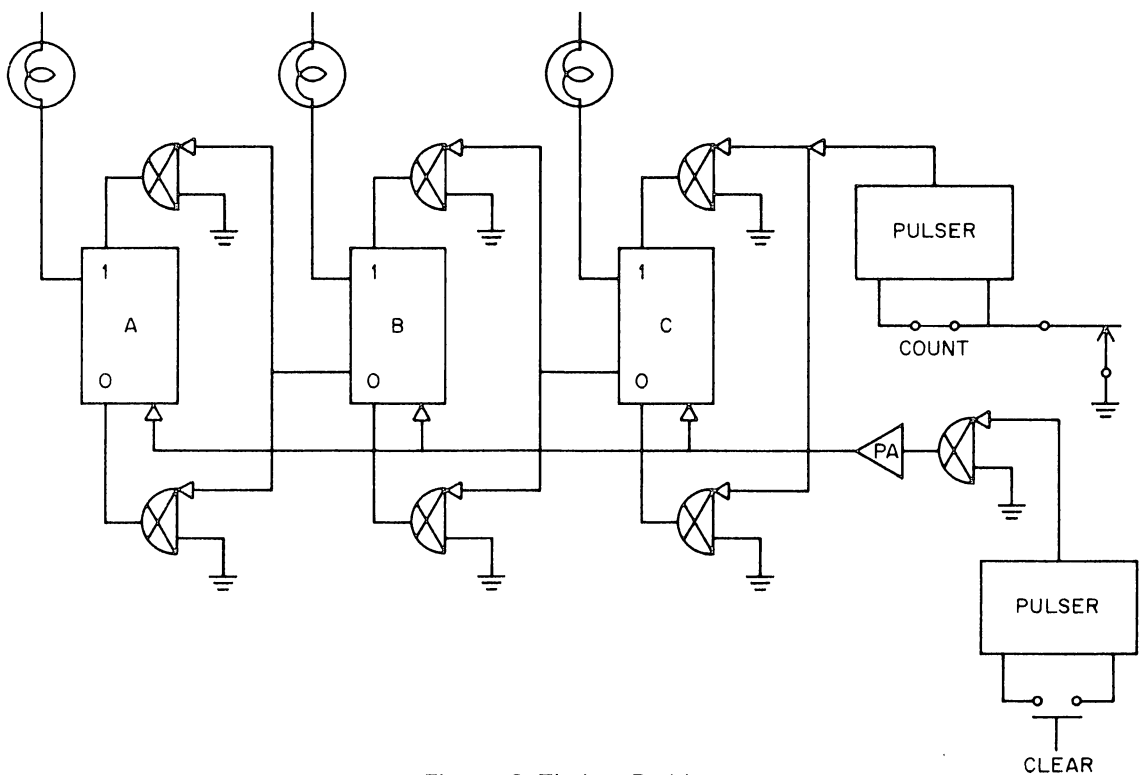


Figure 2 Timing Problem

2. To see how the override signal eliminates this problem, remove the pulse amplifier from the circuit as in Figure 3. Test the clearing action on each of the 7 states.

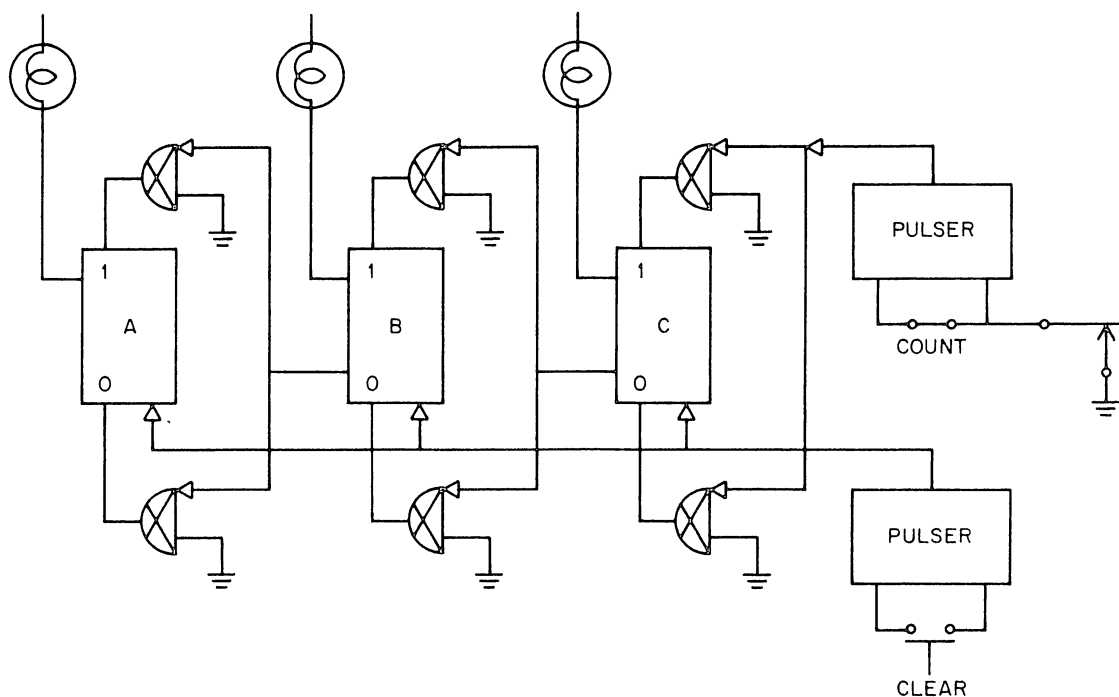


Figure 3 Override with a Push-Button

If the counter is operating at high frequencies, a clear signal of several milliseconds, such as that generated by the push-button, would take far too much time. However, a signal of about 400 nanoseconds duration will be sufficient to override all carries and still be short enough for fast operation. The delay one-shot and similar circuits are used to generate such a signal. Figure 4 shows how this can be done. The circuit also includes an inverter, since the output of the one-shot is negative for the duration of the delay and must be inverted to provide the correct polarity for driving the clear input.

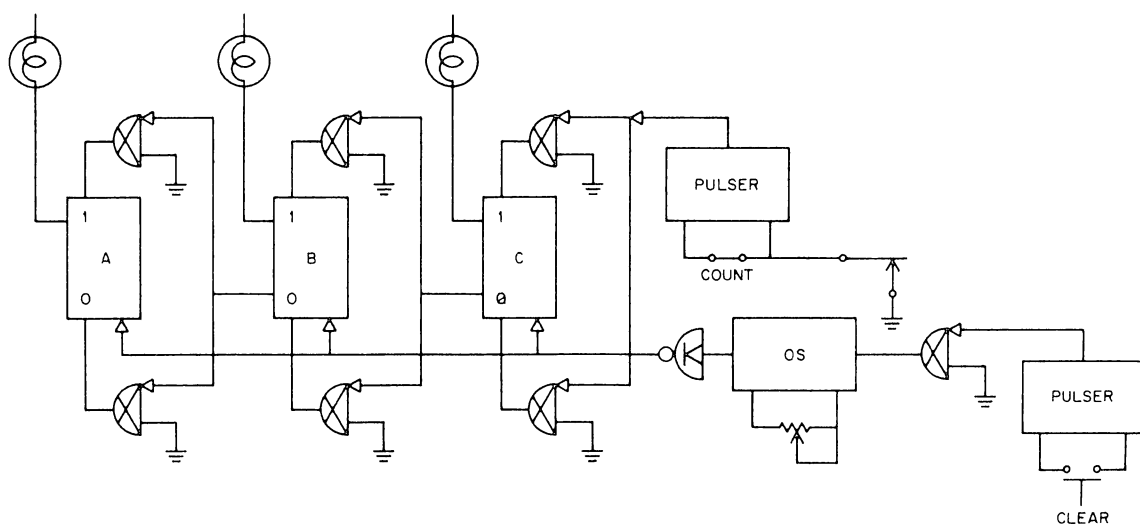


Figure 4 Override with a Delay One-Shot

3. Construct the circuit of Figure 4 and test it by clearing out each of the 7 possible numbers.

Correct clearing every time can also be guaranteed by inhibiting the carry gates during the clear signal, as illustrated in Figure 5. The level inputs of all the DCD gates are brought to a common line which is connected to a toggle switch. When the toggle switch is closed the level inputs are grounded (enabled). Before a clear pulse is given, the switch may be opened, thus disconnecting the level inputs from ground and inhibiting all carry gate signals. The gate is then reclosed before counts enter.

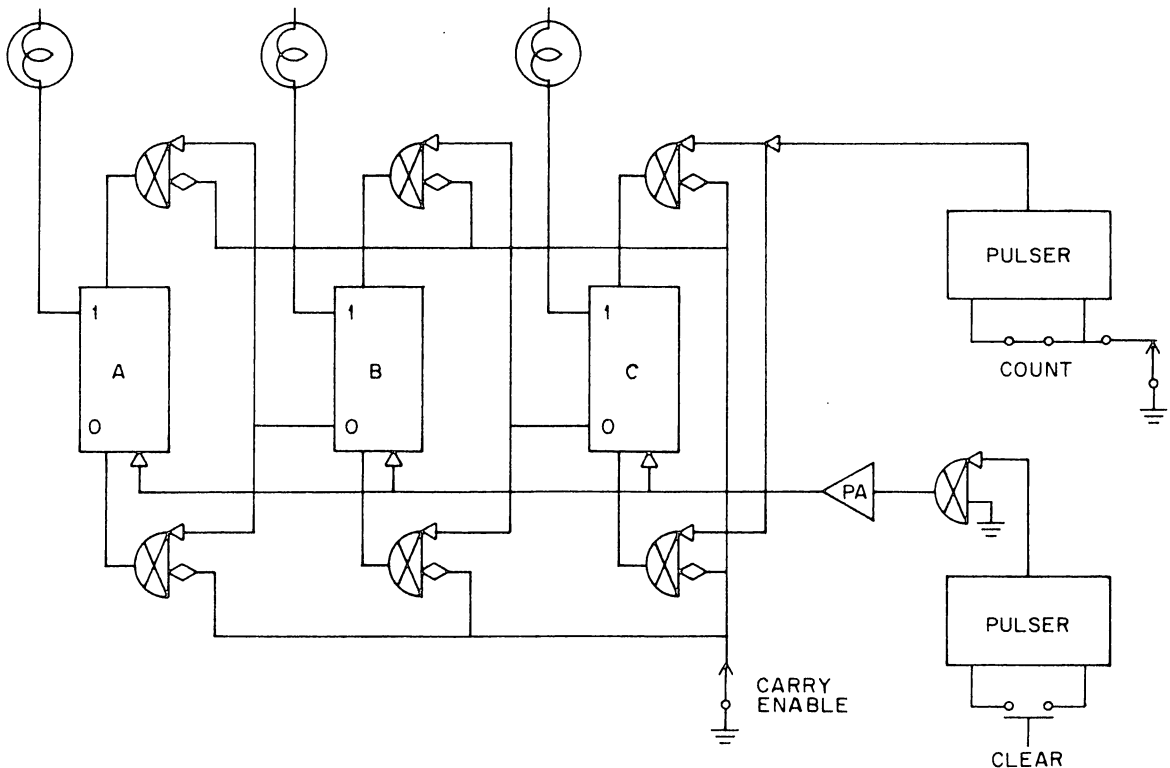


Figure 5 Inhibit

4. Modify your circuit as shown in Figure 5, then try clearing out each of the 7 possible numbers.

PART 2 RANDOM SIGNALS

If two signals are coming into the same circuit at random times with respect to each other, then a simple inhibit or override is not sufficient to guarantee correct operation. Suppose, for example, that you have pulses from two sources, P1 and P2. Even if P1 is a much longer pulse than P2, so that it would normally override P2, occasionally a P2 pulse could enter the circuit just at the end of the P1 override. In these cases the override might be sufficient in some circuits and insufficient in others, depending on the sensitivity of each circuit.

To guarantee correct operation of the logic circuit at all times, random input signals should be synchronized: that is, they should be forced to occur only at fixed times.

One signal should be designated as the primary signal. This may be the highest frequency input or it may be simply an arbitrary clock circuit that is used to generate high frequency standard pulses. All inputs are synchronized to the primary signal, so that they occur only at the same time as the primary pulses. Once the signals are synchronized, the time of their occurrence is known, so override and inhibit techniques can be used reliably. Or the synchronized inputs can be sent through delays of different durations so that they arrive at the main circuit at convenient times.

PART 3 THE SYNCHRONIZER

Basically, the synchronizer is made by ANDing together the random input signal and the primary pulses. The results of this AND gate are used to set a single flip-flop. If the random signal arrives at the same time as the primary pulse, then the single synchronizer flip-flop decides to accept or reject the input.

Figure 6 shows a command level synchronizer which synchronizes a randomly changing logic level to a train of primary clock pulses. It generates output pulses on one of two lines depending on the state of the input logic level. If the logic level is in the process of changing at the time a clock pulse occurs, the output may occur either at A or at B. However, it will never occur at both points; nor will it ever be split, with half of the pulse appearing at A and half appearing at B.

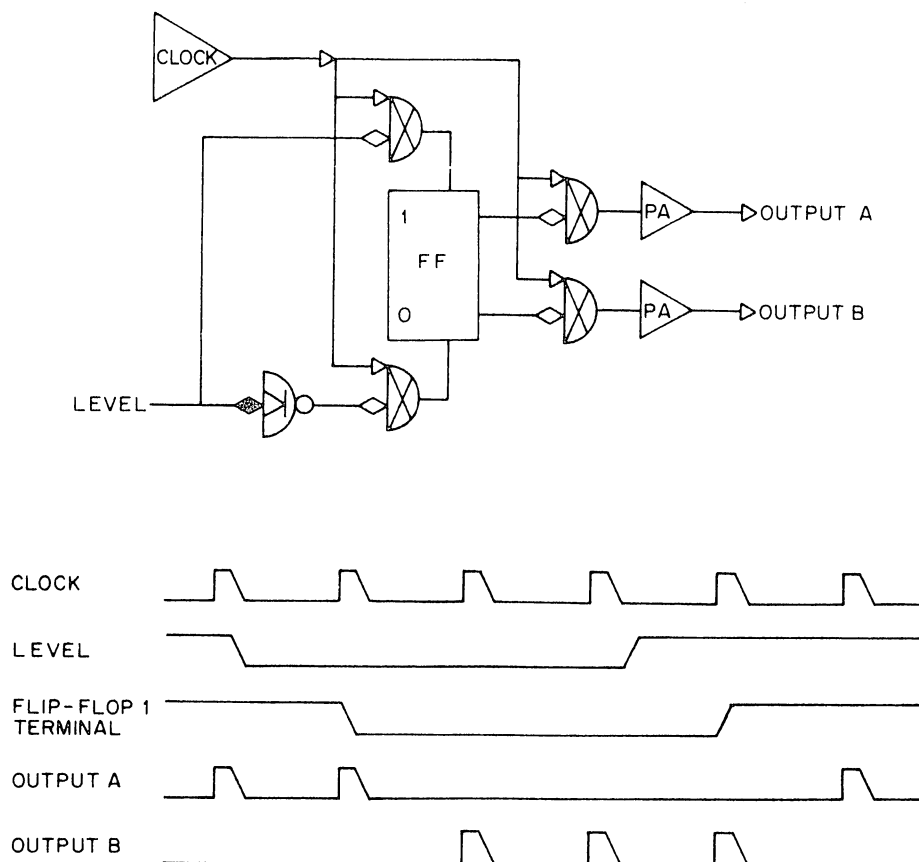


Figure 6 Level Synchronizer

Figure 7 shows a start-stop synchronizer. In this circuit, start and stop pulses are converted into a randomly changing level by flip-flop A. This level is then converted into a synchronously changing level by flip-flop B. Finally, the output pulse amplifier gates out pulses only when flip-flop B is in the ONE state.

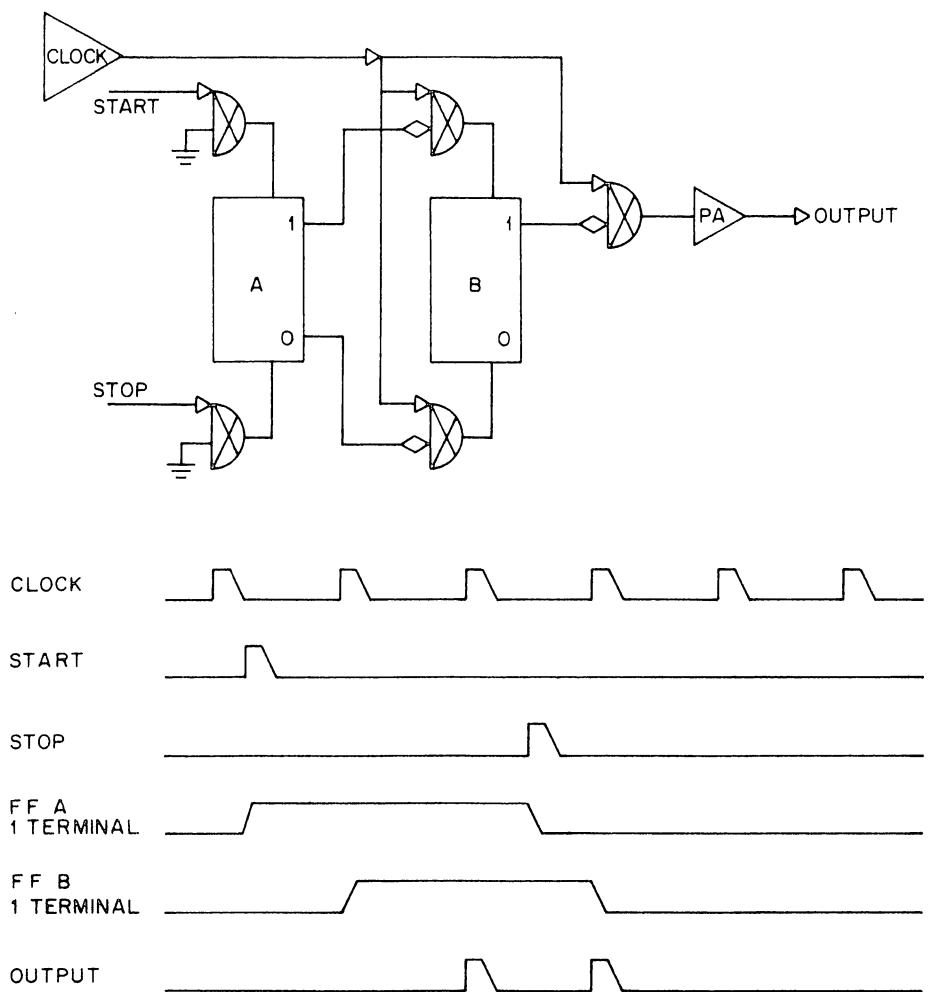


Figure 7 Start-Stop Synchronizer

Figure 8 shows a method for converting a single asynchronous trigger pulse into a single synchronous trigger pulse. The random trigger is converted to a random level by flip-flop A. This is converted to a synchronously changing level by flip-flop B. The next clock pulse will generate an output which resets both flip-flops. The output pulses are delayed by at least one clock period with respect to the trigger pulse. Since the output also clears flip-flop A, the clock frequency must be more than twice the trigger pulse frequency.

PART 4 USE OF THE SYNCHRONIZER

As an example of a circuit with two random inputs, consider a ring counter (Figure 9) where a single ONE is being circulated among a ring of flip-flops by a high frequency clock. To start the operation of the ring counter, a reset signal must put one flip-flop in the ONE state and all others in the ZERO state. But since the clock is continuously producing shift pulses, if the reset occurs at the same time as the shift, an error may result.

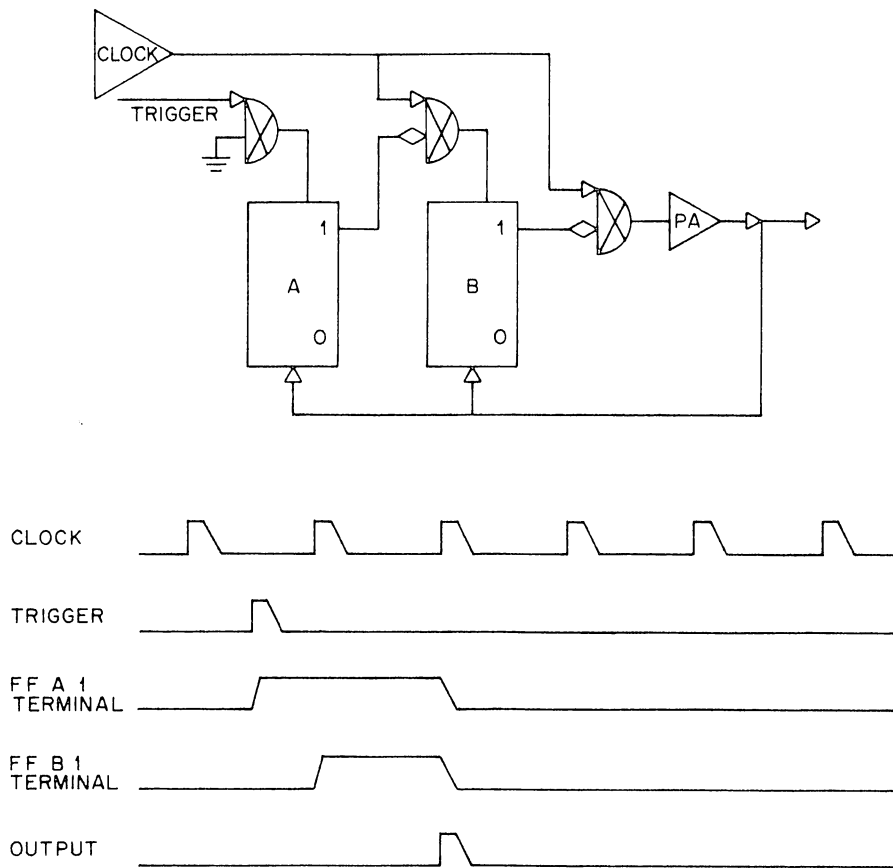


Figure 8 Pulse Synchronizer

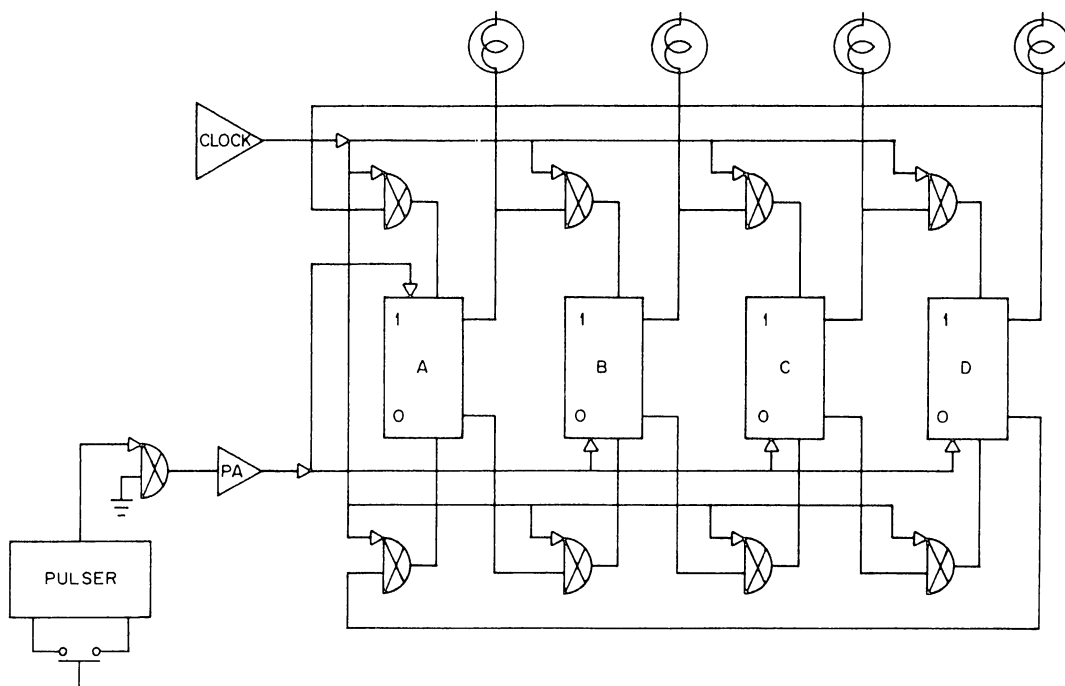


Figure 9 Ring Counter with Random Inputs

Notice the difference between the ring counter circuit of Figure 9 and counter circuit of Figure 1. In the counter circuit, there was no problem with count inputs and clear signals overlapping since they were generated separately by push-buttons. The signals which caused difficulties were extraneous carries formed by the clear. However, being generated by the clear, these signals always occurred at the same time as the clear and could be completely eliminated with an override or an inhibit.

In the ring counter, the reset does not generate any extraneous signals. However, the shift pulses, produced independently by the clock, may appear at the same time as the reset. It is in this situation, when both just happen to appear at the same time, that the difficulty arises. An override is not sufficient, an error might still result if the end of the override should happen to coincide with the shift pulse.

The odds that a given reset pulse will cause an error depends on the timing characteristics of the circuitry and the frequency of the shift pulses. The Logic Laboratory circuits operate at 2 megacycles. If a minimum of 500 nanoseconds is allowed between pulses, the circuits will operate correctly even under worst case conditions.*

Of course, the equipment is not usually operated under worst case conditions; however, in designing or testing the circuit, you should always assume that it will at some time be operated under these conditions.

If the shift frequency is greater than 1 megacycle, then it will be impossible to allow 500 nanoseconds for the shift action plus 500 nanoseconds for the reset action between any two shift pulses. Every reset action would be expected to fail.

At a shift frequency of 500 kilocycles, there will be 2 microseconds between shifts. A failure would be possible if the beginning of the reset pulse occurred within 500 nanoseconds after the beginning of the shift pulse or within 500 nanoseconds before the beginning of the shift pulse. In either case, there would not be a full 500 nanoseconds between pulses. Thus, for every shift pulse, there is 1 microsecond during which the reset must not occur. The chances of having a failure are calculated by dividing the illegal time (1 microsecond) by the total time (2 microseconds). Thus at 500 kc the chances of failure are $\frac{1}{2}$.

For an arbitrary shift frequency, f , the period between pulses will be $p = 1/f$. The illegal time will still be 1 microsecond, so the chances of a failure will be $1\mu \text{ sec}/p\mu \text{ sec}$. At 100 kilocycles, p equals 10 microseconds and the odds are 1/10. At 10 kilocycles the odds are 1/100, etc.

When random inputs are at high frequencies and errors are likely to occur, the need for a synchronizer is obvious. When the error may occur only one time in a thousand, or one time in ten thousand, or perhaps one time in a hundred thousand, the question arises as to whether or not a synchronizer is really necessary. The answer is most emphatically, yes.

Consider a computer which is operating at a frequency of 1 megacycle and where the average problem being run on this computer will require $\frac{1}{2}$ hour. There are one million operations performed per second, sixty million per minute and 1,800 million operations during the course of a single problem. If an error occurred only once in every million operations, then almost none of the problems would be run correctly.

*This interval is normally measured from leading edge to leading edge. When a pulse of greater than 500 nanoseconds is used on a direct input terminal then the only requirement is that the following pulse does not begin before the longer duration pulse is ended.

An error that occurs this infrequently is not easy to detect. One's hand would certainly get tired long before he had pushed a button that many times. Detecting such an error on the oscilloscope would be even more difficult. For this reason, a test system must be cleverly devised so that the circuit will trap itself when it makes an error.

If the equipment is part of, or attached to, a general purpose computer, then the test is normally a program written so that the computer performs the same operation repeatedly, tests itself for the correct results, and types out an error message if the result is not correct.

If the equipment is part of a special purpose computer, then a stock of spare modules is usually kept for test purposes. The equipment is set to repeatedly perform its standard function and the outputs (or other points in the circuit) are monitored with a comparator, a parity checker, or some similar type of logic network constructed for this purpose.

5. Construct the circuit of Figure 9 and observe the output of one of the flip-flops on the scope.

- (a) At a clock frequency of 500 kc, count the number of times that you have to reset the circuit until you have caused a failure 5 times. From your count, calculate your average chance of failure.
- (b) Repeat part (a) with a clock frequency of 200 kilocycles.
- (c) The chances of failure that you calculated from your observations are probably less than those calculated from theoretical considerations. Why?
- (d) Add a reset synchronizer to your circuit as shown in Figure 10. Can you now run the clock at 2 megacycles without observing any failures on the reset?

PART 5 SPECIAL PROBLEM

6. Figure 11 shows a circuit which scales an input frequency by an arbitrary number as set in the toggle switches.

The counter starts at 0001. With each successive pulse input, it counts up until it reaches the number set in the toggle switches. This is detected by the diode AND gate which inhibits the carry gates and enables the pulse amplifier. The next input pulse will be gated through the pulse amplifier to generate the output signal and to reset the counter to the number 0001.

- (a) Construct the circuit in Figure 11 and test it by operating it from the push-button. Then connect the clock input and simultaneously observe the clock and output waveforms with the switch register set to various numbers. Sketch the waveform when the switches are set to 1010. (Use a dual-trace oscilloscope or add the waveforms as outlined in Appendix B.)
- (b) Disconnect the carry enable line from the diode gate and permanently ground it. With the switches set to 1010 observe and sketch the clock and output waveform. If the circuit does not divide by ten as it should, observe the various flip-flop outputs and see if you can explain what is happening in the circuit.
- (c) Disconnect the carry enable line from the gate and permanently ground it. Modify the circuit using the delay one-shot to generate the reset signal so that the counter will operate correctly with any arbitrary number in the switch register.

7. If the chances of a failure are very low, such as 1 chance in 1000 or 1 chance in 10,000, then many thousands or millions of tests must be performed to detect the error. A semiautomatic test set-up for the ring counter can be constructed as shown in Figure 12. Diode gates are used to detect the states $A'B'C'D'$ or BD or CD. When any of these states occur, an error flip-flop is set. This lights an indicator and also halts the shift register at the end of the next shift pulse. With this test set-up, the reset input can be activated at a very high frequency and the error will still be detected.

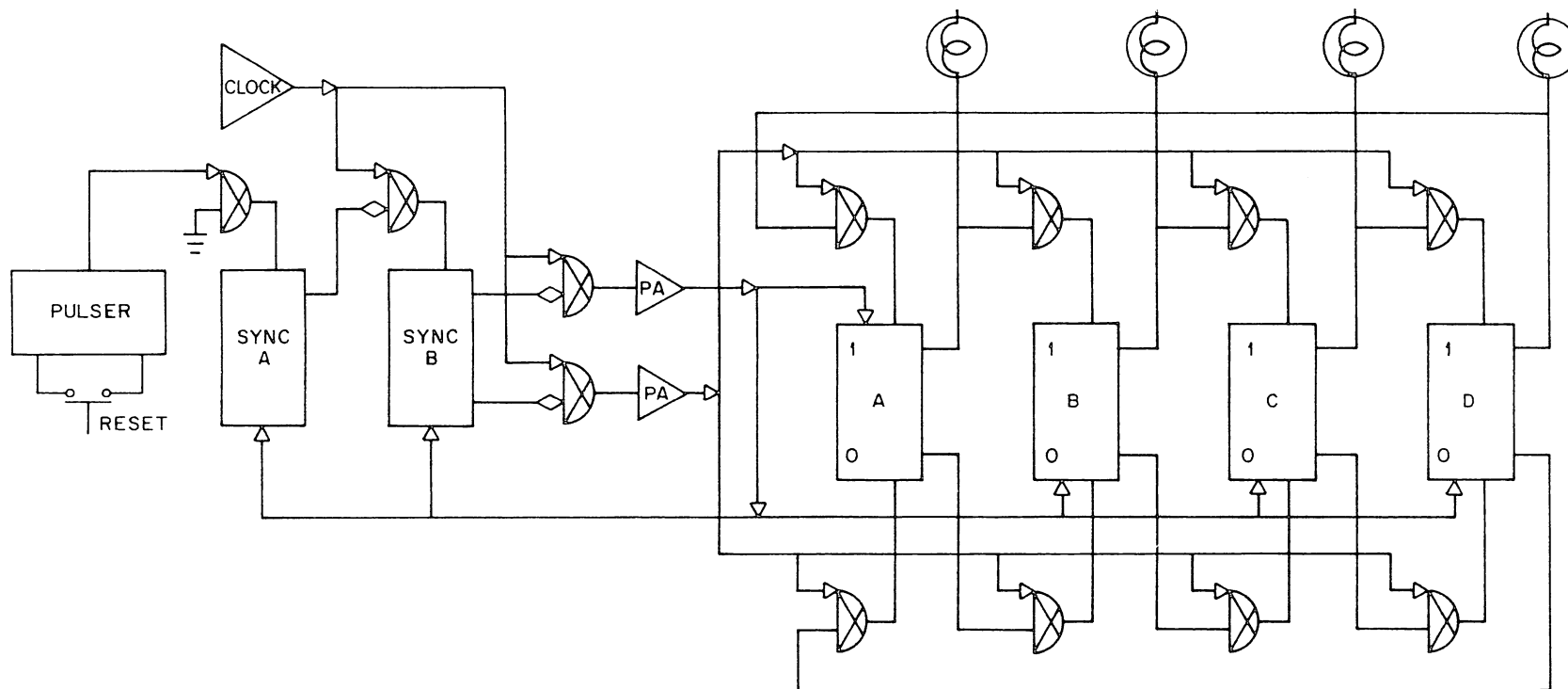


Figure 10 Ring Counter with Inputs Synchronized

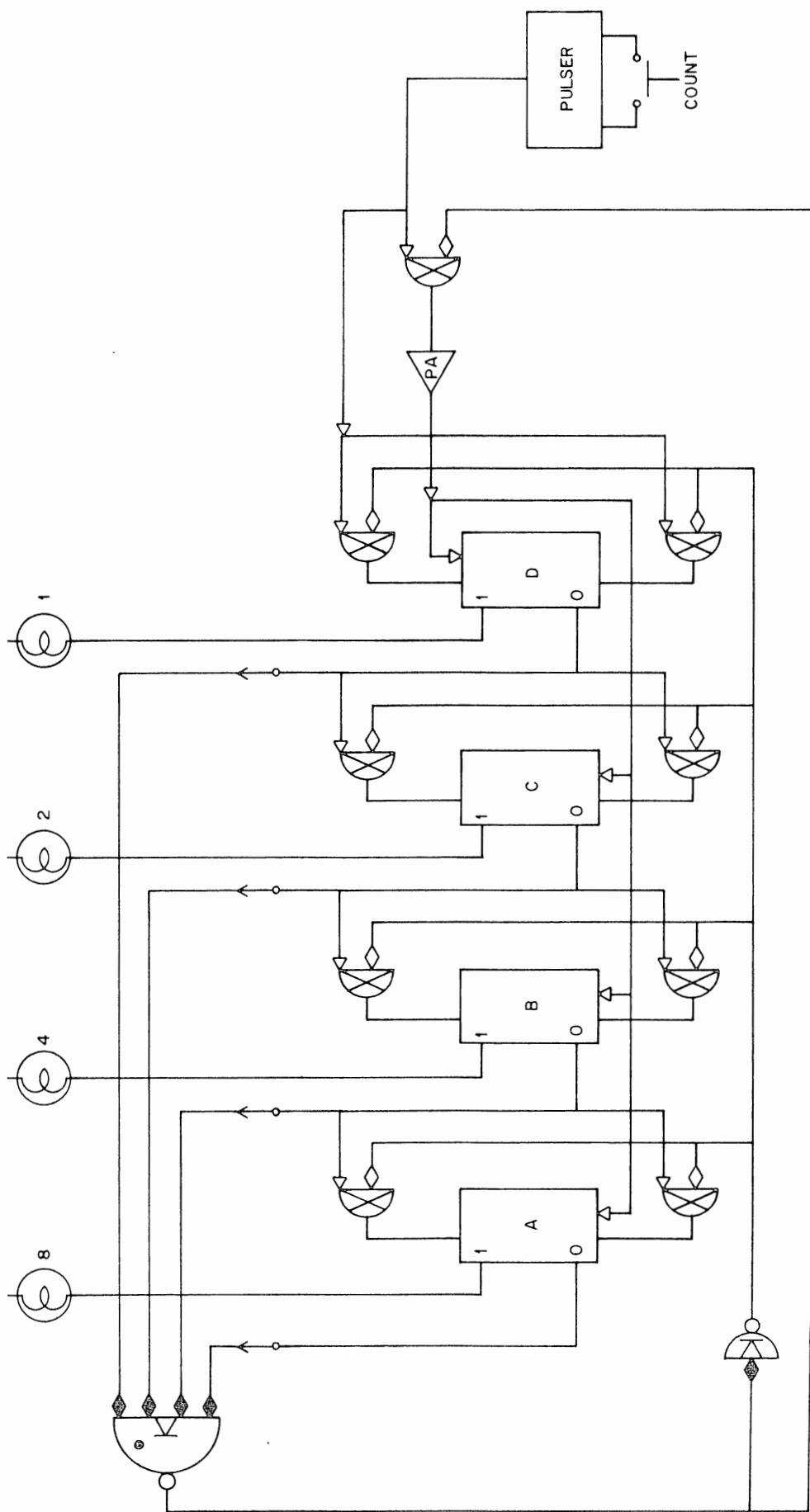


Figure 11 Frequency Scaler (Counter)

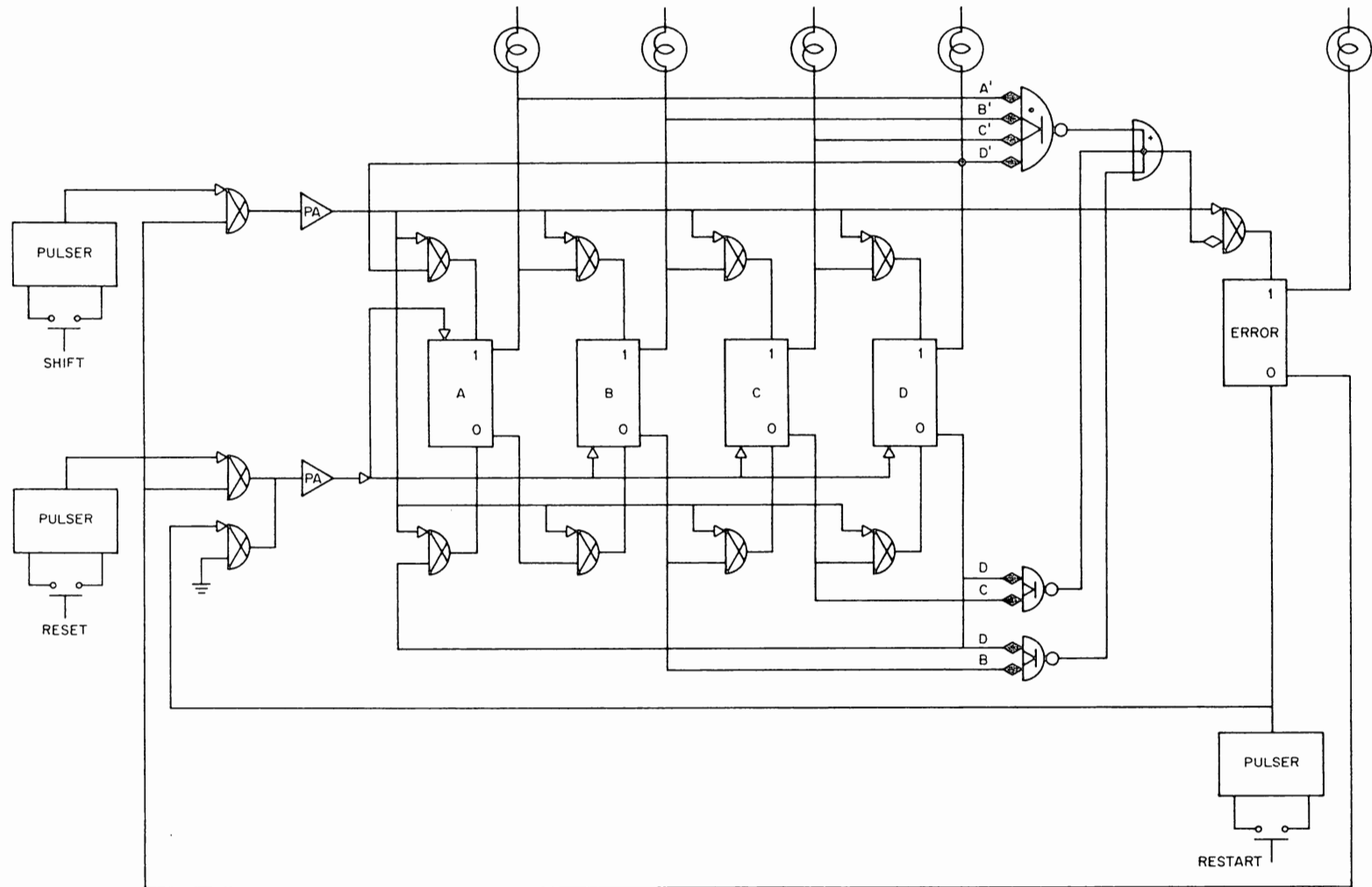


Figure 12 Ring Counter with Tester

To truly measure the odds, the reset signal should be completely random. Since such a signal is hard to come by in a laboratory, a fixed frequency may be used, provided its period is not a multiple of the shift period. This means that the reset signal should be generated separately, and neither frequency should be a harmonic of 60 cps.

Figure 13 shows how the reset can be produced from an external signal generator (either square wave or sine wave). The generator will produce a bipolar signal. This is converted into a Logic Laboratory standard signal with a Schmitt trigger circuit. (These circuits normally form the pulsers for the push-buttons. One of them should be removed and plugged into the mounting panel for use here.)

- (a) Wire the circuit of Figure 12, and test it, using push-buttons for both the shift and reset inputs.
- (b) Remove the pulsers from the reset input and wire the circuit of Figure 13. Set the signal generator for 3 to 10 volts peak (6 to 20 volts peak-to-peak) and check that a square wave is produced at the pulser output.
- (c) Remove the push-button connections from the shift and reset inputs of your ring count. Connect the clock to the shift input and the external signal (as shown in Figure 13) to the reset input. Set the clock for 50 kc and the signal generator to 70 cycles. Using a watch with a second hand, measure the average time to failure on five trials.
- (d) Calculate the average chances of failure from your observations. Remember that there are 70 resets per second.
- (e) Repeat (c) and (d) at a clock frequency of 5 kc.
- (f) Repeat (c) and (d) at a clock frequency of 500 cycles.
- (g) From the theoretical considerations in Part 4 calculate the expected chances of failure for shift frequencies of 50 kc, 5 kc and 500 cycles. Are these much higher than your observations? Why?
- (h) Show that the gating network $A'B'C'D' + BD + CD$ will detect all possible errors.

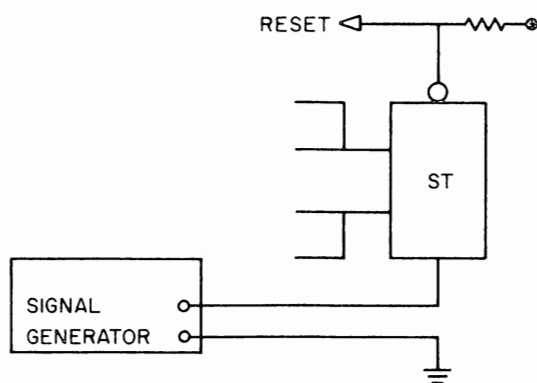


Figure 13 External Reset Logic

EXPERIMENT XIII

INTRODUCTION TO ANALOG-DIGITAL CONVERSION

PART 1 CONVERTER USES

Analog-to-digital converters (ADCs) and digital-to-analog converters (DACs) are basic to many computer input-output channels. ADCs measure signals from thermometers, pressure gauges, flowmeters, resistance bridges, photocells, microphones, etc. Through DACs, a computer can drive meters, motors, and loud speakers; it can control ovens, draw graphs on an oscilloscope, and perform many other useful functions automatically.

Converters take many forms. For example, the deflection of a speedometer needle in an automobile represents an analog value which you convert to digital when you read the number corresponding to its position. This section discusses the types of converters most commonly used with a computer—those which convert between an electrical analog signal and a binary number.

PART 2 DIGITAL-TO-ANALOG CONVERSION

The basis of a digital-to-analog converter is a simple resistor network such as the circuit illustrated in Figure 1. Each switch represents a binary bit. If switch D is closed, while all the others are open, the current at I will be $V/8R$. If switch C is closed, while all the others are open, the current at I will be $V/4R$, or twice as much as the current due to closing switch D. Similarly, if only B is closed, the current will be $V/2R$, or twice the current due to switch C, and four times the current due to switch D. Each switch produces double the current of the previous one, just as each binary bit has double the weight of the previous one. If more than one switch is closed, the individual currents will add together. In this way, the total current will always be proportional to the binary number contained in the switches.

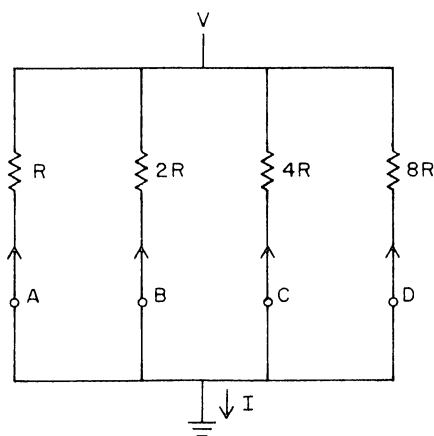


Figure 1 Binary Weighted Summing Network

The Logic Laboratory A-D panel contains a resistor-switch network, similar to that of Figure 1. The switches are semiconductor, rather than mechanical elements. They are closed when a ground signal is applied, and are open when there is a negative input or no connection.

The DAC is followed by an amplifier which increases the signal range and also converts it from a current signal into a voltage signal. The output of the amplifier is at ground when all the DAC inputs are off (open circuit or negative voltage). If one or more of the switches are on (grounded), the amplifier output is then a negative voltage which is proportional to the binary weights of the inputs. The rheostat controls the feedback of the amplifier, allowing the gain (or output voltage range) to be adjusted.

1. Connect the DAC inputs to toggle switches and the output to a meter as illustrated in Figure 2. Adjust the gain so that the output is -4 volts when the input is 1000.

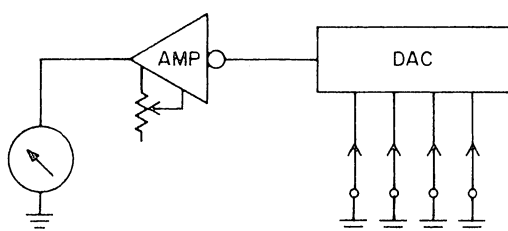


Figure 2 DAC Connections

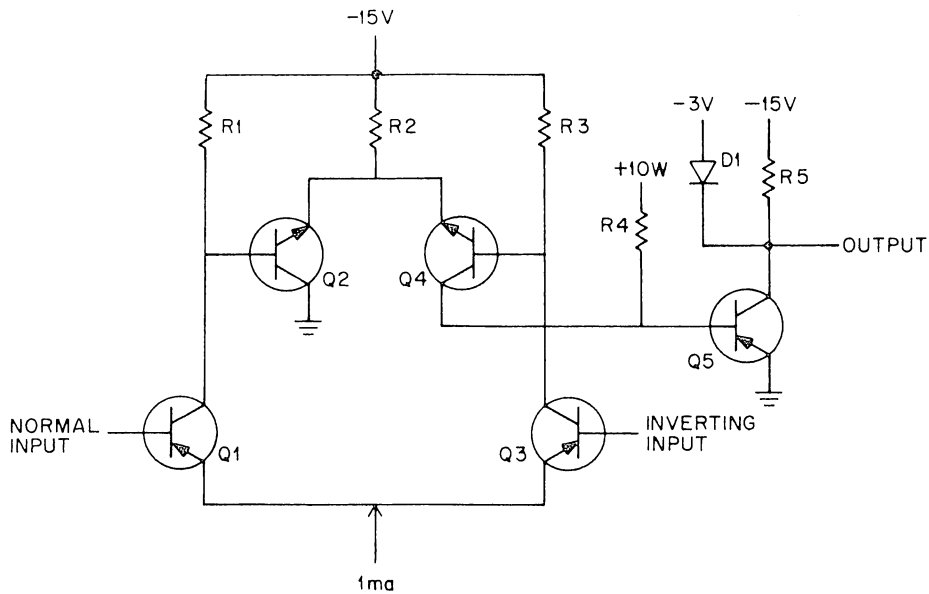
- (a) What is the output voltage corresponding to each of the possible input numbers?
- (b) What is the weight of each?
- (c) What is the voltage increment between successive binary numbers?

PART 3 ANALOG-TO-DIGITAL CONVERSION

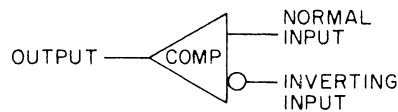
The comparator is the basic circuit used in analog-to-digital conversion. This circuit compares two voltages and produces a binary output indicating which of the two signals is more negative.

Figure 3 shows the logic diagram and a simplified circuit schematic for the comparator in the Logic Laboratory A-D panel. From the schematic you can see the operation of the circuit. It is a two-stage differential amplifier. If the normal input is more positive than the inverting input, then the majority of the current from the 1 ma current source will be routed through transistor Q3. This will bring the base of Q4 more positive than the base of Q2 and current will flow through R2, Q4, and R4, bringing the base of transistor Q5 more negative than its emitter. This causes transistor Q5 to conduct and its output will go to ground.

If, on the other hand, the inverting input is more positive than the non-inverting input, then most of the current will pass through Q1. Very little current will go through Q3, so that the base of Q4 will be more negative than the base of Q2, and Q4 will conduct little or no current. The resistor R4 will hold the base of transistor Q5 more positive than its emitter and the transistor will act as an open circuit. In this case the output voltage will be determined by R5 and D1, and will be approximately -3 volts.



CIRCUIT



SYMBOL

Figure 3 Comparator

The comparator circuit, then, gives a binary decision as to which of two voltages is larger. The digital equivalent of an analog voltage can be found by applying the unknown signal to one input of the comparator and using a digital-to-analog converter to try different voltages at the other input of the comparator until the correct voltage is found.

2. To see the operation of the comparator, wire the circuit of Figure 4. The potentiometer connected between the negative voltage and ground will serve as an analog input signal which can be monitored by the nearby meter. When this is connected to the inverting input of the comparator and the DAC is connected to the non-inverting input, the comparator output indicator will show which of the two voltages is greater. When the light is on, the analog signal is more negative than the DAC voltage. When the light is off, the analog signal is more positive than the DAC input. With the DAC again adjusted so that the binary number 1000 equals -4 volts, find the smallest number that turns the indicator off for each of the following voltages:

- (a) -3.3 volts
- (b) -7.2 volts

- (c) -4.7 volts
- (d) -6.3 volts

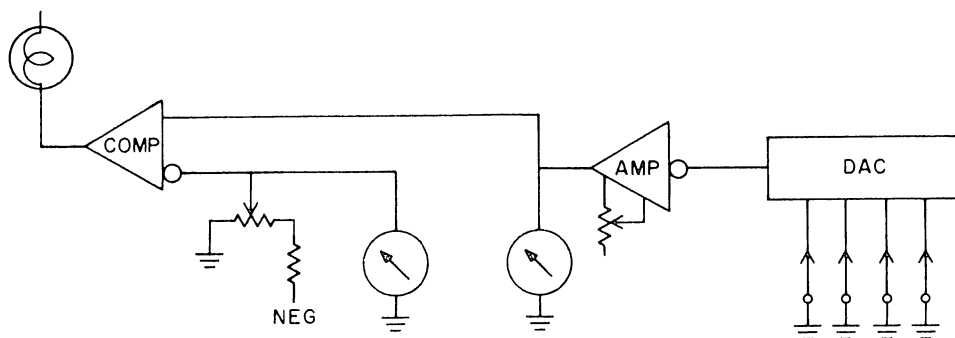


Figure 4 Observing the Comparator Operation

If the DAC is driven by a counter as illustrated in Figure 5, then the different binary numbers may be tried in sequence. To convert an analog voltage to its digital equivalent, reset the counter, then push the count button repeatedly until the comparator indicator goes off. The number contained in the counter will then be the binary equivalent of the input analog voltage.

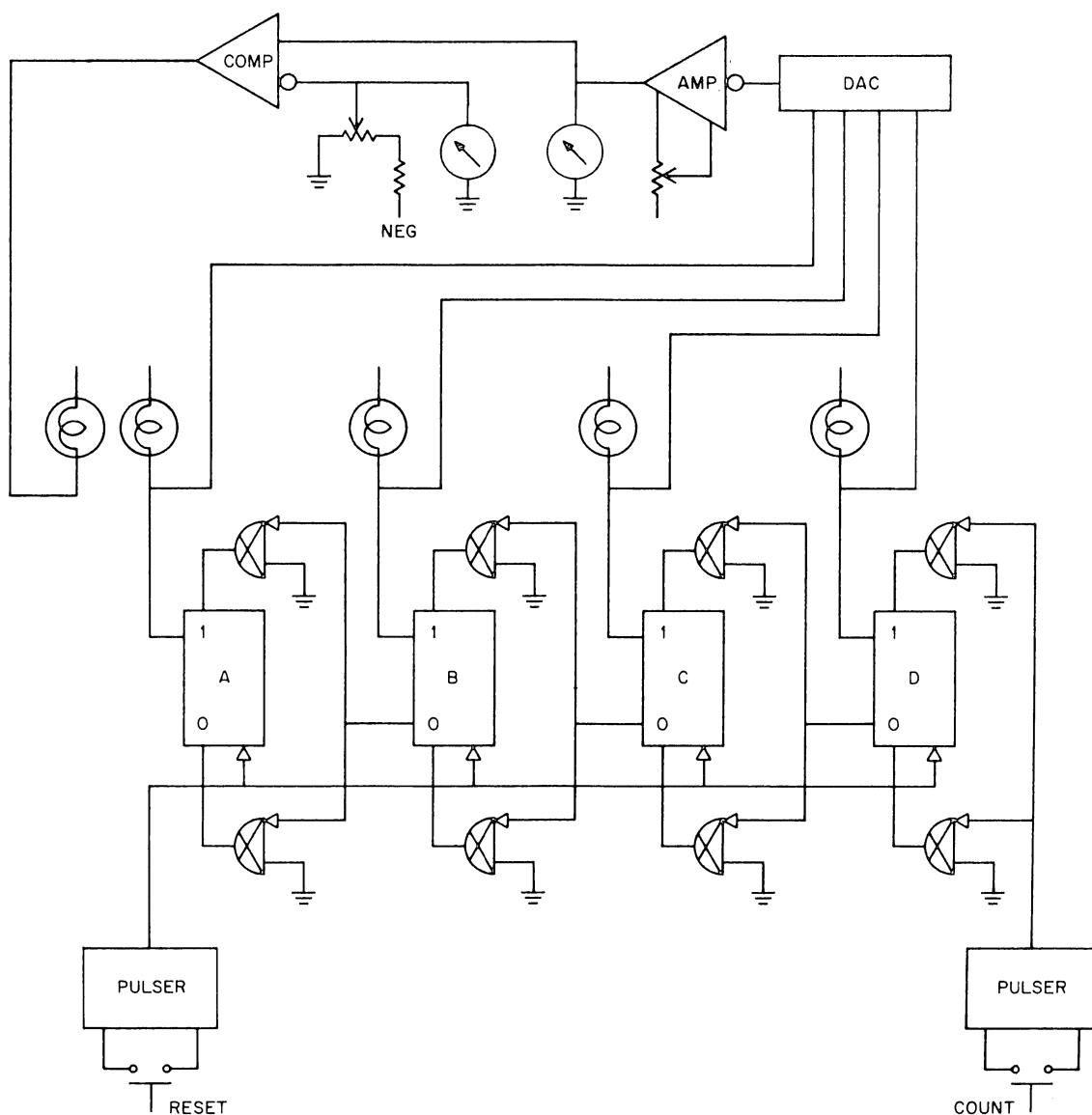


Figure 5 Trying Binary Numbers in Sequence

3. Construct the counter circuit of Figure 5, and convert the following voltages:

(a) -1.3 volts

(c) -4.7 volts

(b) -3.7 volts

(d) -6.3 volts

To further automate the conversion, the comparator output can be applied to the input gates on the counter so that count pulses are inhibited as soon as the DAC voltage exceeds the input voltage. The count pulses can then be generated by two delay one-shots connected in a loop as shown in Fig. 6.

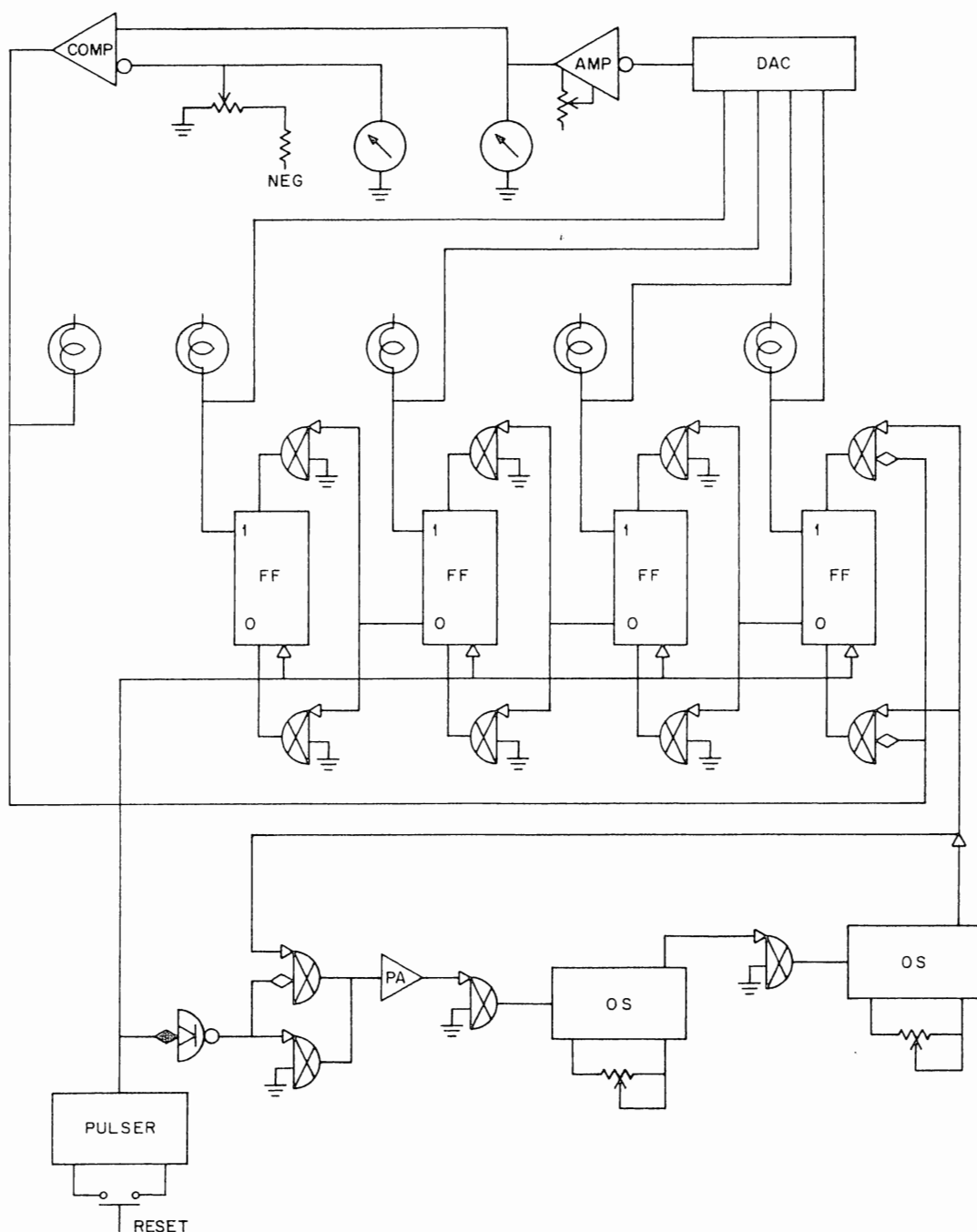


Figure 6 Counter Converter

The action is both initiated and stopped by the push-button. When the button is depressed, it resets the counter. When it is released, it starts a pulse circulating around the delay line loop. The next time it is depressed, it will inhibit the delay line loop as well as resetting the counter.

4. Connect the comparator feedback and delay one-shots to your converter as shown in Fig. 6. Set the delays for 1 microsecond each. Convert the following voltages:

- | | |
|------------------|------------------|
| (a) -1.3 volts | (c) -4.7 volts |
| (b) -5.7 volts | (d) -6.3 volts |

This, then, forms a complete converter. A single signal from the push-button starts the converter. The rest of the process is automatic. This type of converter is commonly called a counter converter.

PART 4 THE CONTINUOUS CONVERTER

The continuous converter is a variation of the counter converter, made by adding gates for down counting as well as up counting. The comparator controls both the up and down count gates, routing pulses into the up gates when the DAC voltage is too low, into the down gates when the DAC voltage is too high.

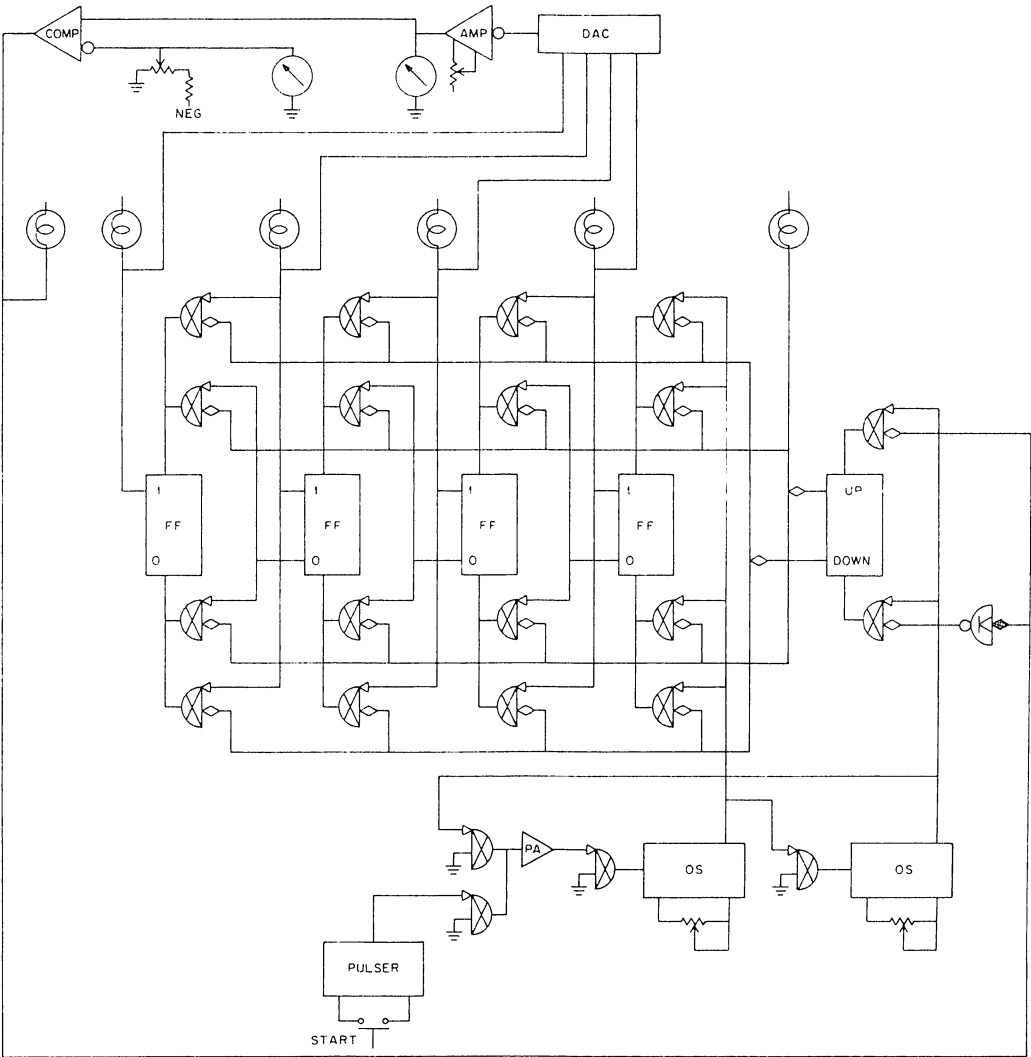


Figure 7 Continuous Converter

When the start signal is given, the count pulses are routed in this manner until the converter output goes to the same voltage as the input signal. Once it is locked on, the continuous converter will follow any changes in the input voltage, producing at all times an instantaneous reading of the corresponding analog value.

5. Figure 7 shows a continuous converter, similar to a counter converter but with down count gates added. A synchronizer flip-flop has also been added to assure that the count enable signals do not change during the counting process. (Such changes could be caused either by noise on the input signal line or by fast feedback from the least significant counter bit, through the DAC, and through the comparator before a counting process has been completed.) Construct this converter and watch how it follows the input voltage as it is increased and decreased.

- (a) When the analog input is left at a constant value, the digital reading will tend to oscillate. Explain what causes this.
- (b) If the analog input exceeds the maximum range of the digital-to-analog converter, the counter will overflow to zero. Design and construct a circuit which eliminates this.

EXPERIMENT XIV

ADVANCED STUDIES IN ANALOG-DIGITAL CONVERSION

PART 1 THE SUCCESSIVE APPROXIMATION CONVERTER

In a typical monitoring and control system, the computer is required to measure and analyze analog inputs from a number of different sources. A separate analog-to-digital converter for each input would be quite costly. It is far less expensive to multiplex all of the signals into one ADC as illustrated in Figure 1. Each input signal is connected to the converter through a switch, either a relay or its semiconductor equivalent. The switches are closed one at a time, while the value of the corresponding input signal is measured by the ADC and the information is sent into the computer. Then the next switch is closed and corresponding value measured, etc.

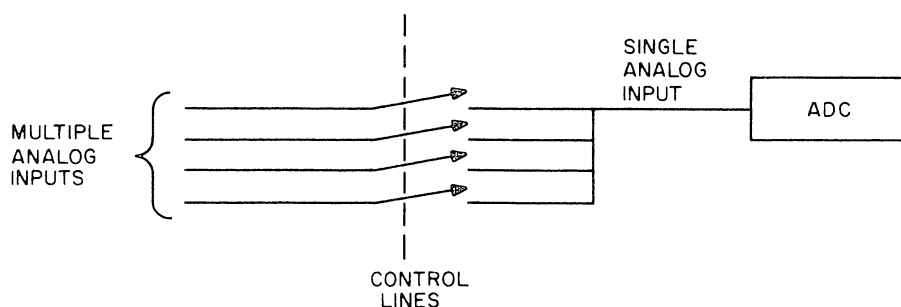


Figure 1 ADC with Multiplexed Inputs

In a multiplexed analog-to-digital conversion system, the ADC must measure each new voltage very quickly in order that all inputs can be sampled often enough to avoid losing information. If the converter were too slow, then any high frequency changes in the input signals would not be received by the computer. To find the value of each new signal as fast as possible, a successive approximation converter is used. This converter is similar to the ADCs in Experiment XIII but slightly more sophisticated. Figure 2 shows a simplified diagram of the basic elements—a DAC, a comparator, a flip-flop register to drive the DAC and control circuitry with a pattern generator. The converter operates by repeatedly dividing the possible input voltage range in half . . . each time determining one binary bit in the output. Each division, or each bit, requires a separate step. At the beginning of the process, the converter must be reset. Thus a 4-bit conversion requires four decision steps, plus one reset step, for a total of five steps, as described below.

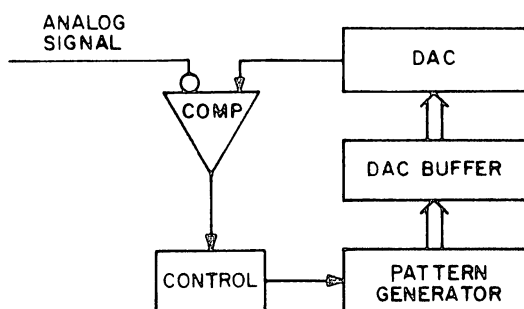


Figure 2 Successive Approximation Converter

Reset. The buffer is initially set to the number 1000 in order to test the most significant bit. This produces -4 volts at the DAC output. If the analog input is more negative than -4 volts, then the most significant bit must be a ONE. The comparator output will signify this by going to ground. If, on the other hand, the input is more positive than -4 volts, the most significant bit should be a ZERO. This is indicated by a negative comparator output.

Step 1. The most significant bit is corrected by clearing the corresponding flip-flop if the comparator is negative, or by doing nothing if the comparator is at ground. The second bit is tested by setting it to a ONE, which produces either -2 or -6 volts, depending on the most significant bit. In either case the logic is the same as in the previous step. For example, assume that the most significant bit is a ZERO. The DAC output will be -2 volts. If the analog input is more positive than -2 , then the second bit should be a ZERO and the comparator output will be negative. If the input is between -2 and -4 , the second bit should be a ONE, indicated by a comparator output at ground level.

Step 2. The second bit is cleared or not, depending on the comparator. The third bit is tested by setting it to a ONE.

Step 3. The third bit is cleared or not and the last bit is set to a ONE.

Step 4. The last bit is cleared or not.

The way in which the converter arrives at different numbers is illustrated in Figure 3. This logic can, of course, be extended on to any number of bits.

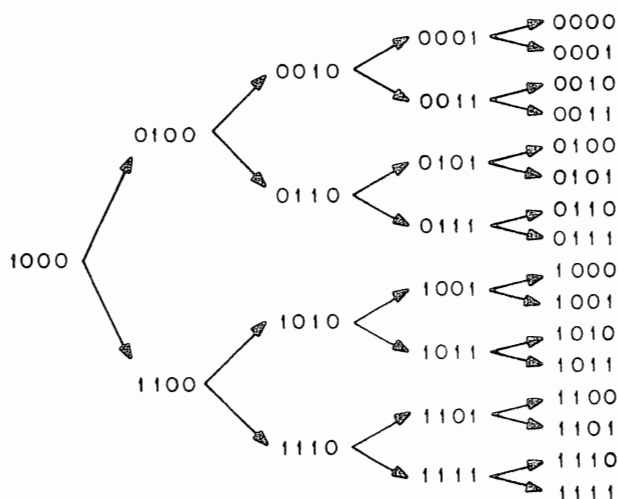


Figure 3 Operation of Successive Approximation Converter

Figure 4 shows the detailed logic for the DAC, the DAC buffer, and the comparator. Notice the separate pulse inputs and how the inputs are gated to make the decisions described above.

Figure 5 shows the control and the three flip-flop pattern generator which produce the reset and step pulses. The flip-flops are connected as a switched tail ring counter which can generate six states:

000
100
110
111
011
001

Only the first five of these are used. At reset time the flip-flops are forced to 000. The first shift pulse will set flip-flop E, generating the step 1 pulse. The second shift will set flip-flop F, generating the step 2 pulse, etc.

1. Construct and test the successive approximation converter as illustrated in Figures 4 and 5. Adjust the amplifier gain as in Experiment XIII.

(a) Converting to -3.2 volts, record the readings from the comparator and all of the flip-flops at the end of each step.

(b) Repeat (a), converting to -3.8 volts.

(c) Figure 6 shows how the control pulses can be generated automatically. The reset pulse, which should be a long pulse to override any extraneous step pulses, comes from a delay one-shot. Replace the push buttons with this automatic control. Set the clock for 500 kc and the delay one-shot for 1 microsecond. Observe the movement of the DAC output on an oscilloscope. With the input at -3.2 volts, observe and sketch the DAC waveform.

(d) With the input at -3.8 volts, observe and sketch the DAC waveform.

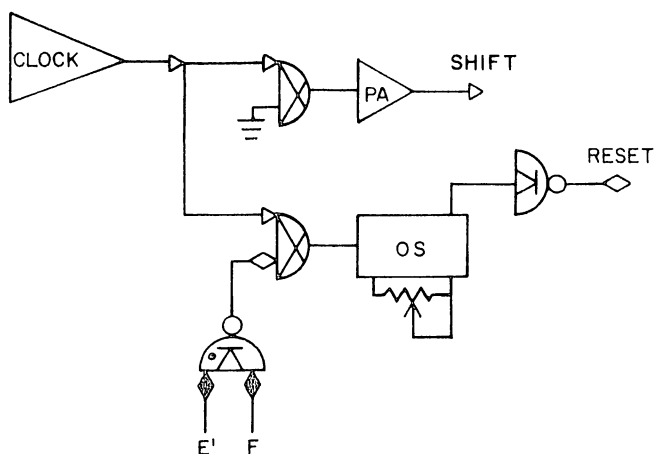
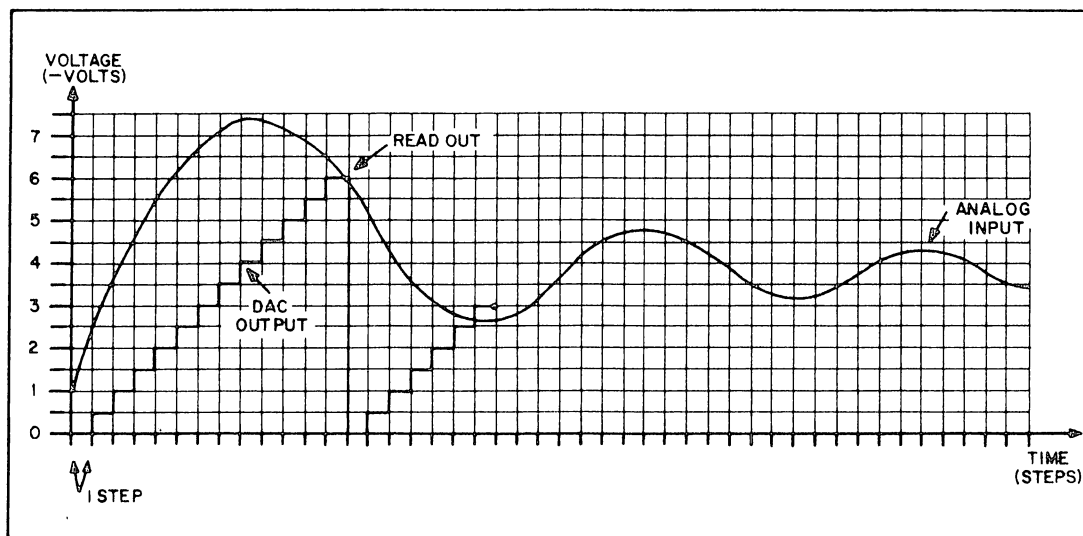


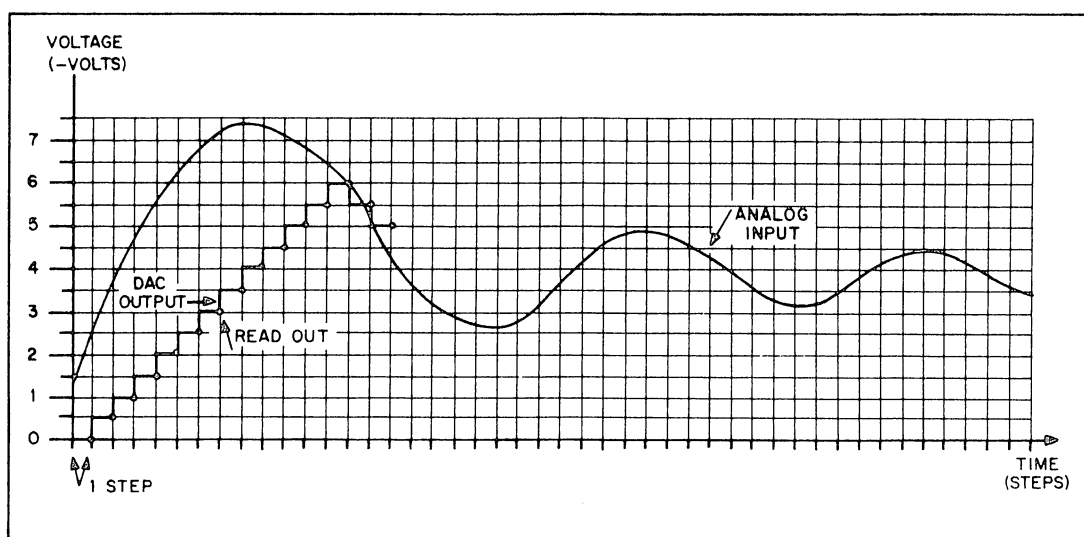
Figure 6 Automatic Generation of Control Pulses

PART 2 COMPARISON OF ANALOG-TO-DIGITAL CONVERTERS

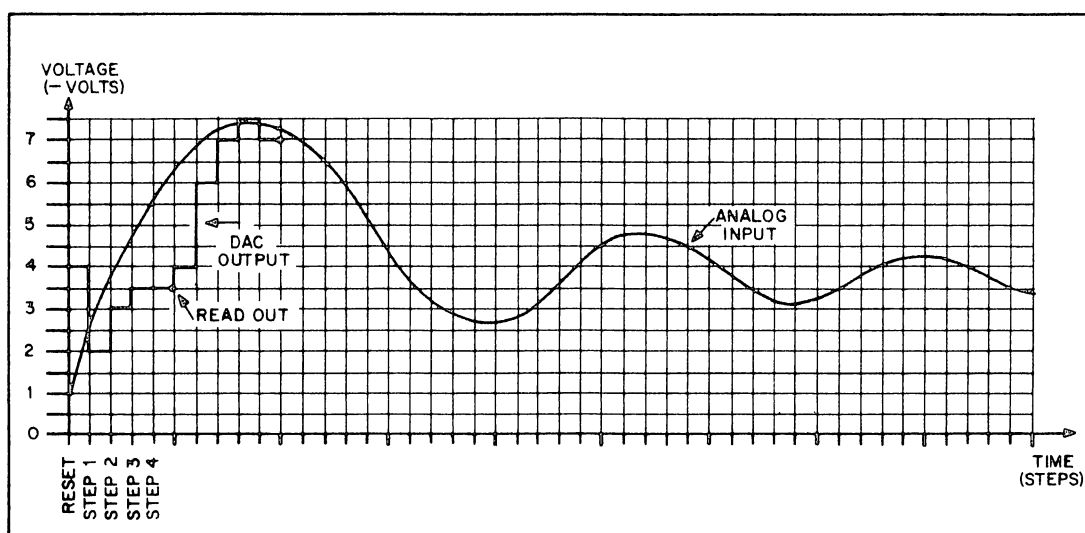
For digital-to-analog conversion, you have studied only one technique. It is fast and straight forward. It completes its conversion in one step, does not require any elaborate control circuitry, and is suitable for most applications.



(a) Counter Converter



(b) Continuous Converter



(c) Successive Approximation Converter

Figure 7 Converter Response to a Transient Signal

Analog-to-digital converters, on the other hand, are awkward, requiring a large number of steps and special control. This awkwardness has led to the design of many types of converters, each with its own advantages and disadvantages.

The counter converter requires the least circuitry. Hence it is the most economical, and it is used whenever speed is not critical, as in digital voltmeters. It is also used in peak reading meters. If the input analog signal dips and rises, the number in the counter will increase whenever the input voltage exceeds the DAC signal. Thus the counter will automatically go to the peak input voltage. The disadvantage of the counter type is, of course, its slowness. It may take up to 2^N steps ($2^N - 1$ counting steps and 1 reset) to convert an N-bit number. For a 4-bit converter, this means only 16 steps, but for a 10-bit converter, it means 1,024.

The continuous converter, once it locks onto the input signal, is the fastest type of ADC, completing a new conversion at every step. Thus it is very good for recording all the details of the analog signal. However, if the input changes faster than the converter can follow, the converter will fall behind until after the input signal has slowed down. The maximum rate of change that the converter can follow is:

$$\frac{V}{2^N \Delta t} \text{ volts per microsecond}$$

where

V = voltage range of the converter (in volts)

N = number of bits

and

Δt = time between count pulses (in microseconds)

The successive approximation converter, although it is the most expensive of the three, is the most widely used because it performs well over a wide range of applications. It is considerably faster than the counter converter, performing a complete N-bit conversion in $N + 1$ steps (including the reset). Its advantage over the continuous converter is that each conversion is independent of the previous one. Thus, it cannot fall behind a fast signal. For these reasons, the successive approximation converter can measure a new voltage, such as one just switched in by a multiplexer, and find the correct result quicker than either of the other two types of converters.

2. Figure 7 (a, b, c) shows a transient signal and how each type of converter would respond if such a transient were applied at its input. This figure also shows how the DAC outputs would appear and dots indicate when readouts would be available. Continue the DAC curves to the end of the transient, then connect the readout dots to show how the input signal would be reconstructed from the digitized values.

(a) Which converter produces the best description of the high frequency portion of the transient?

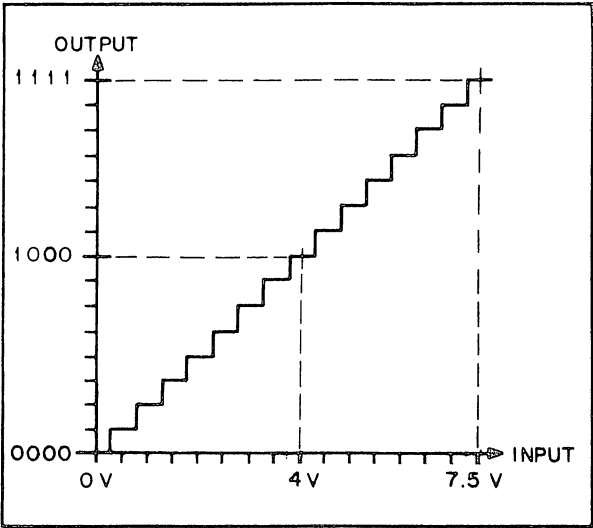
(b) Of the low frequency portion?

PART 3 ACCURACY

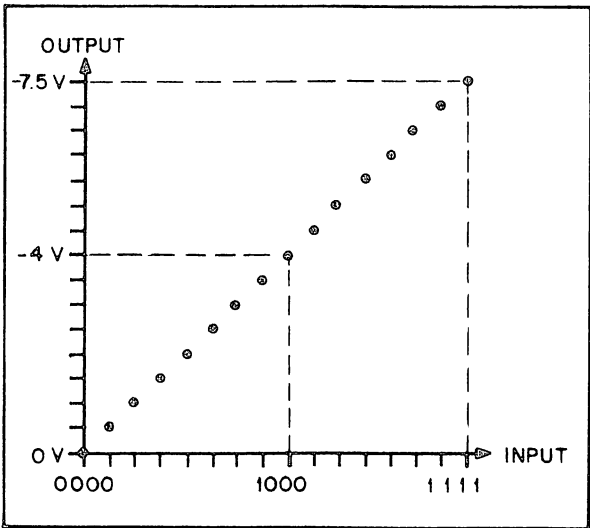
Since the end result of conversion is to represent a given value in different terms, it is important to know how accurate the representation is. There are three types of errors which can occur in a A-D conversion: quantization error, noise, and DC or switching-point error.

Quantization error occurs in analog-to-digital converters only. It is simply a part of the conversion process. In fact, whenever any continuous signal is quantized, there is a quantization error which is equal in magnitude to the smallest quantum. For example, if a yardstick, with marks every eighth inch, is used to measure a table, there will be a possible error of $\frac{1}{8}$ inch. Similarly, in a binary coded analog-to-digital converter, the least-significant bit is the smallest increment of measure and the largest possible increment of error. In most systems the error is centered so that it is between $+\frac{1}{2}$ a bit and $-\frac{1}{2}$ a bit. (Written as $\pm\frac{1}{2}$ LSB.)

A digital-to-analog converter accepts only as many bits as it can convert. For this reason, the accuracy measurements are made only on that set of discrete points which are used in the converter. Thus, in a DAC there is no quantization error. Figure 8 shows output versus input plots for both ADCs and DACs. Notice how the ADC plot forms a staircase while the DAC plot is a set of discrete points lying along a straight line.



(a) ADC



(b) DAC

Figure 8 Output vs. Input

Noise in a converter is due to incomplete filtering of the 60 cycle power or pickup from external high frequency signals. In a digital-to-analog converter, noise appears as a variation in the output signal. In an analog-to-digital converter, it affects the switching point (the point at which the converter stops producing one output and starts producing the next higher or next lower number). It causes the switching point to vary slightly so that the same input voltage will not always produce the same output number.

3. To study the effects of noise, construct the circuit of Figure 9. Set the input to 1111. For DAC measurements, monitor the output of the amplifier with an oscilloscope. For ADC measurements, watch the output of the comparator indicator. (It is not necessary to construct the entire ADC converter since only the DAC and the comparator affect the accuracy. The final results for any type of converter can be predicted by watching the comparator output.)

(a) Observe the DAC output and measure the noise on the oscilloscope. (The noise will be predominantly 60 cycles and approximately 10 millivolts in amplitude.)

(b) To see the effect of the noise on an analog-to-digital converter, adjust the input to -7.5 volts then vary the voltage slowly. There will be a narrow region over which the indicator appears to be only dimly lit. This is when the comparator is switching back and forth because of the noise. If you were running the comparator output into a flip-flop buffer, the results would sometimes read 1111 and sometimes 1110.

(c) To study the effects of pick-up, set the clock to its maximum frequency. Then take a long lead from the clock output and wrap it seven or eight times around the lead between the DAC output and the comparator input. Again measure the noise from the DAC output as in part (a). This time the frequency of the noise will be near 2 megacycles.

(d) With the clock lead still wrapped around the DAC output lead, vary the analog voltage as in part (b) and notice the flickering of the indicator. If possible, measure the input voltage range through which the flickering occurs.

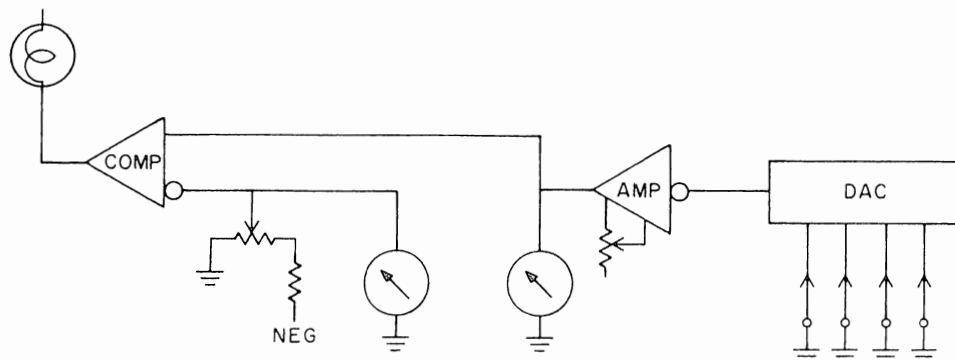


Figure 9 DAC-ADC Test Set-up

DC or switching point errors are due to (1) resistors in the divider network not being quite the correct value, (2) the impedance of the transistor switches which drive the divider network not being exactly zero when closed or exactly infinite when open, and (3) variations in β of the transistors. These factors are constant over a relatively short period of time, although they vary with time, temperature changes, humidity, power fluctuations, etc. They cause slight errors in the average DAC output, and they cause the switching point of an ADC to occur at a slightly incorrect voltage.

4. The DC error of the converter can be measured using the circuit of Figure 9. Since the noise will be considerably smaller than the DC error, it may be neglected.
- (a) Adjust the gain on the DAC amplifier so that 1000 produces -4 volts.
 - (b) Measure the DAC output voltage for each of the 16 possible input numbers. Compare these with the theoretical voltage and determine which is the largest error and where it occurs.
 - (c) To measure the switching point accuracy of the ADCs, set the toggle switches to 0000. Then, starting at 0 volts, increase the analog input until the indicator goes off (or is blinking). This is the point where the ADC would stop converting to the number 0000 and begin converting to 0001. Now set the toggle switches to 0001 and look for the switching point between 0001 and 0010. Continue this procedure to 1111. Prepare an output versus input plot. Draw the ideal curve (a straight line from the origin to -7.5 volts, 1111), then draw an ideal curve which takes quantization error into account (a staircase with the switching points at the odd quarter volts). What is the maximum ADC error with respect to the ideal curve? What is the maximum ADC error with respect to the ideal staircase?
 - (d) The accuracy of DAC is usually quoted as a percentage of the full scale. Considering full scale as 8 volts, what is the accuracy of your DAC?
 - (e) The accuracy of an ADC is usually quoted as a percentage of the full scale $\pm \frac{1}{2}$ LSB. With full scale again 8 volts, what is the accuracy of your ADC?

EXPERIMENT XV

COMPUTER DESIGN

PART 1 COMPUTER ELEMENTS

A digital computer has four main elements. You have studied each of them separately: the arithmetic element, the memory, the control logic, and the input-output. Now you will have a chance to put them together and make a complete digital computer.

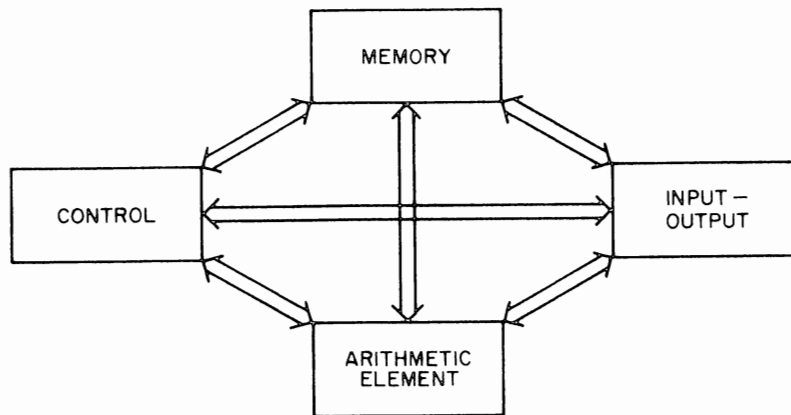


Figure 1 Basic Elements of a Computer

The arithmetic element is the heart of the computer where calculations are performed. It consists primarily of flip-flops and gates, although pulse amplifiers and other circuits are sometimes used to provide high driving power.

The memory element stores data not being operated upon by the arithmetic element and it may also store instructions. In a typical computer system, the memory consists of magnetic core stacks, often augmented by mass storage devices such as magnetic tape systems, drums, or discs. In the Logic Laboratory, the memory consists entirely of flip-flops. Although the unit cost of flip-flops is much higher than bits in a core memory or mass storage device, the overall cost of a small flip-flop memory such as we will construct is considerably less, and can be readily made of standard digital modules.

The need for the input-output section is fairly obvious. A computer would be useless unless it could receive the raw data and make known the results of its computation. The computer may communicate with a human operator by means of a typewriter, cathode ray tube, plotter, microphone and loudspeaker, meters, console lights, switches and buttons, or special contrivances which simulate other devices (such as a simulated airplane cockpit). A computer can exchange information with other computers via magnetic tape and punched paper tape. Through analog-digital converters, the computer can receive information from external instruments such as flow meters, pressure meters, or thermometers, and send out control signals to motors, valves, relays, displays, etc.

The control element is usually the most sophisticated. It must generate a series of pulses, at the correct times and in the right sequence, to control all of the com-

puter's operations. It also initiates the operation when the console start button is depressed and halts the computer when the calculation is complete.

PART 2 GENERAL VERSUS SPECIAL-PURPOSE COMPUTERS

Computers are usually divided into two classes. The general-purpose computer, which can be programmed without hardware modifications to perform a nearly unlimited number of different tasks, and the special-purpose computer, which can perform at most a few variations on one task.

In the earliest general-purpose computers, programming flexibility was made possible by bringing the control points out to terminals on patchboard, similar to those in the Logic Laboratory. The user would then patch various timing and control elements together to generate the proper sequence of operations for the calculation. Today, however, general purpose computers are almost universally of the stored program type. Each possible operation (such as add, shift, check for zero) is represented by a binary code. To prepare a program, the appropriate codes are stored in sequential locations in memory. When the computer is in operation, these instructions are brought one by one from the memory element into the control element. Here the binary number is decoded and pulses are sent out to perform the desired operation. In order to handle these instructions, the general-purpose computer must have a large memory and a fairly complex control.

In the special-purpose computer, on the other hand, the sequence of operations is always the same. This sequence can be governed by a fixed pattern generator in the control circuitry, thus making the control much smaller than that of a general-purpose computer. The memory, which is used only for data storage, may also be small. If only a little data is required, the large scale memory may be eliminated entirely and the remaining requirements filled with a combined memory-arithmetic element.

The advantage of the general-purpose computer is its flexibility. Almost all of the largest computers are general-purpose, so that they can be shared by many users working on different problems. Even if there is only one calculation to be done, the general-purpose computer runs less chance of becoming obsolete, since the program can always be changed to solve a different problem.

The special-purpose computer is faster than the general-purpose computer since it does not need to bring instructions from memory and interpret them. But, its primary advantage is in systems requiring little or no data storage. Here the large scale memory element can be replaced with a combined memory-arithmetic element for a considerable cost saving. Thus, special-purpose computers tend to be used for smaller tasks, such as simple process control, automatic testing of manufactured articles, and converting information from instruments into engineering units.

The computer which you will build in this experiment is a special-purpose computer designed to calculate square roots. This is the type used in instrumentation systems to convert to engineering units. The one difference between your computer and instrumentation computers is that your input will be manual, whereas the instrumentation system usually receives the input directly from the measuring device.

PART 3 THE ALGORITHM

The methods used to do a calculation by hand are not necessarily the most efficient

techniques to use on a computer. The usual method of finding a square root is to try a number, see if it is too large or small, then try another number, and so forth. The square root computer however, will use as its basis an old theorem from number theory; namely, that the sum of the first N odd integers ($1 + 3 + 5 + \dots + [2N - 1]$) is N^2 . For example:

$\begin{array}{r} 1 \\ 3 \\ 5 \\ 7 \\ \hline 16 = 4^2 \end{array}$	or in binary:	$\begin{array}{r} 1 \\ 11 \\ 101 \\ 111 \\ \hline 10000 = 100^{10} \end{array}$
--	---------------	---

To find the square root, this process is simply reversed. Start with the number for which the root is to be found and subtract successive odd integers until the result goes to zero. Counting the number of subtractions gives the square root.

For example:	$\begin{array}{r} 16 \\ -1 \text{ first subtraction} \\ \hline 15 \\ -3 \text{ second subtraction} \\ \hline 12 \\ -5 \text{ third subtraction} \\ \hline 7 \\ -7 \text{ fourth subtraction} \\ \hline 0 \end{array}$ <p style="text-align: center;">the square root is 4 (or 100)</p>	$\begin{array}{r} 10000 \\ -1 \\ \hline 1111 \\ -11 \\ \hline 1100 \\ -101 \\ \hline 111 \\ -111 \\ \hline 0 \end{array}$
--------------	--	---

Since some input numbers will not be perfect squares (that is, they will not go exactly to zero), it is easiest to continue subtraction for all numbers until the result is negative (or overflows). In this case, the root is one less than the total number of subtractions performed. For example (arrows mean overflow):

$\begin{array}{r} 13 \\ -1 \\ \hline 12 \\ -3 \\ \hline 9 \\ -5 \\ \hline 4 \\ -7 \\ \hline -3 \end{array}$	$\begin{array}{r} 1101 \\ -1 \\ \hline 1100 \\ -11 \\ \hline 1001 \\ -101 \\ \hline 100 \\ -111 \\ \hline \leftarrow 11100 \end{array}$	$\begin{array}{r} 16 \\ -1 \\ \hline 15 \\ -3 \\ \hline 12 \\ -5 \\ \hline 7 \\ -7 \\ \hline 0 \\ -9 \\ \hline \leftarrow 110110 \end{array}$
$\begin{array}{r} 17 \\ -1 \\ \hline 16 \\ -3 \\ \hline 13 \\ -5 \\ \hline 8 \\ -7 \\ \hline 1 \\ -9 \\ \hline \leftarrow 110111 \end{array}$	$\begin{array}{r} 10001 \\ -1 \\ \hline 10000 \\ -11 \\ \hline 1101 \\ -101 \\ \hline 1000 \\ -111 \\ \hline 1 \\ -1001 \\ \hline \leftarrow 110111 \end{array}$	
root 3 11	4 100	4 100

PART 4 HARDWARE

The hardware for finding a square root by the odd integer method is quite easy to construct. There must be a square register to hold the input number and perform the subtractions. The odd integers may be generated with a counter where the count pulses enter the second bit instead of the least significant bit.

The control circuitry must initially set the odd integer register to 1 and read the input number into the square register. It then generates pulses to subtract, check for overflow, and, if no overflow is generated, advance to the next odd integer and repeat the cycle. When the overflow is detected, the computer halts.

Since the more significant bits of the odd integer register are receiving one pulse per subtraction, these flip-flops will also keep track of the number of subtractions and generate the root. This is done simply by reading the contents of all odd integer register flip-flops except the least significant bit.

Figure 2 shows a block diagram of the computer hardware. The square register receives the input number and also functions as the subtractor. The odd integer register generates both the odd integers and the roots. It consists of two parts. The most significant bits are a counter. The least significant bit is always a ONE, since it is only used for generating odd integers and all odd integers end in a ONE.

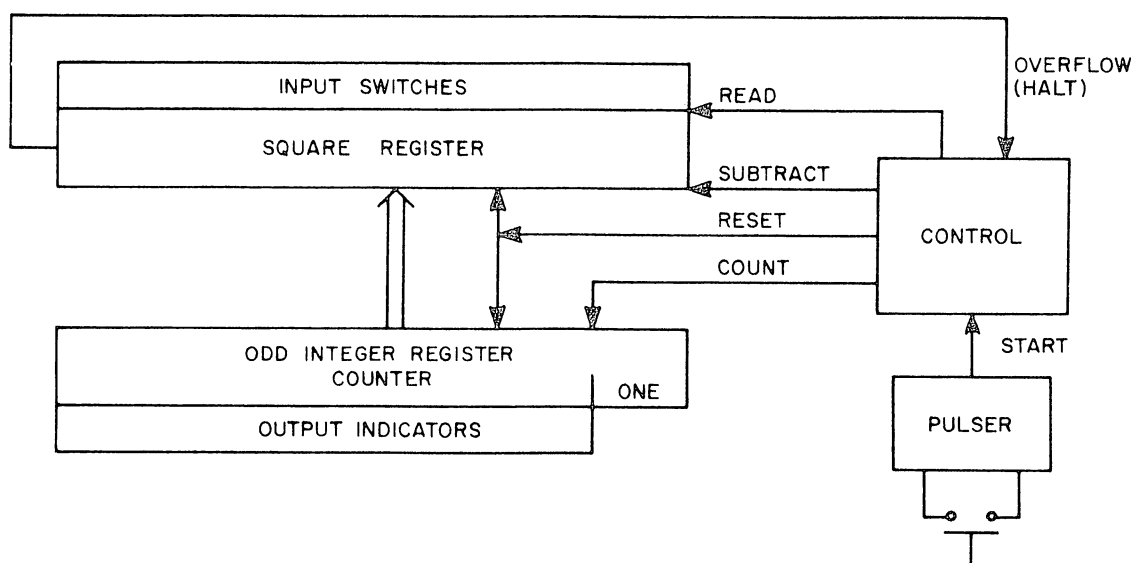


Figure 2 Block Diagram

How does this block diagram relate to the general block diagram of a computer shown in Figure 1? The square register and odd integer register form a combined arithmetic-memory element. They receive raw data from the toggle switches and make known the result via the indicator lights. The control element is the same here as in Figure 1.

How large should the arithmetic element be? The square register must be long enough to contain the largest input number. The counting section of the odd integer register must be as long as the largest possible square root, which is half as long as the square.

Sketching this out, as in Figure 3, shows some simplifications that can be made. The least significant bit of the odd integer register can be eliminated. Since it is always a ONE, it can be replaced with a permanently wired signal. The more significant bits of the square register do not receive inputs from the odd integer counter, so all they need is borrow propagate logic. This is really the same as a down counter — so the more significant bits of the subtractor are merely a down counter. The logic is thus reduced to that of Figure 4.

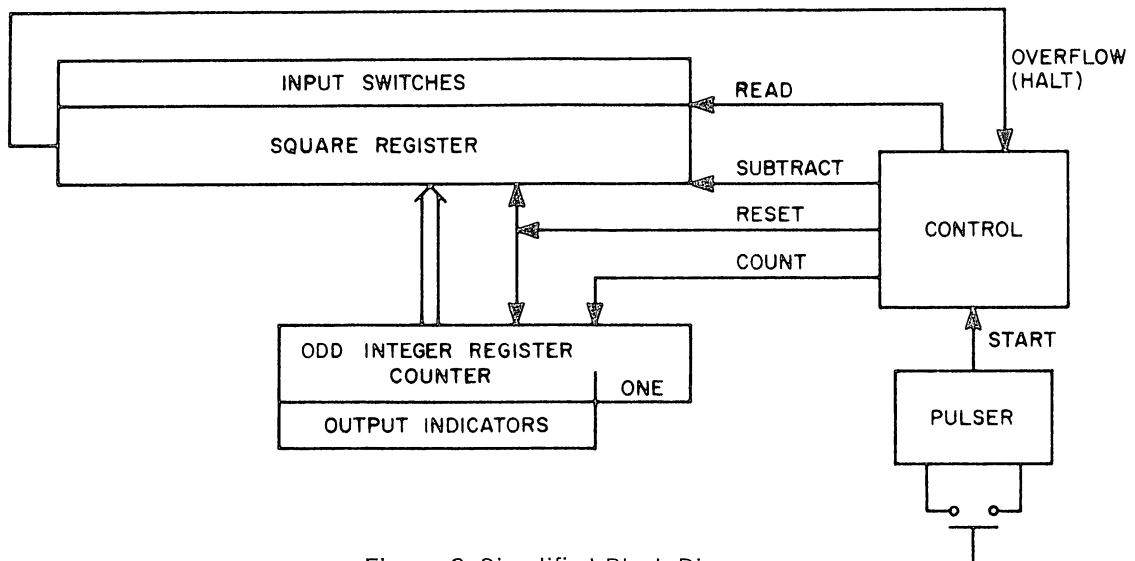


Figure 3 Simplified Block Diagram

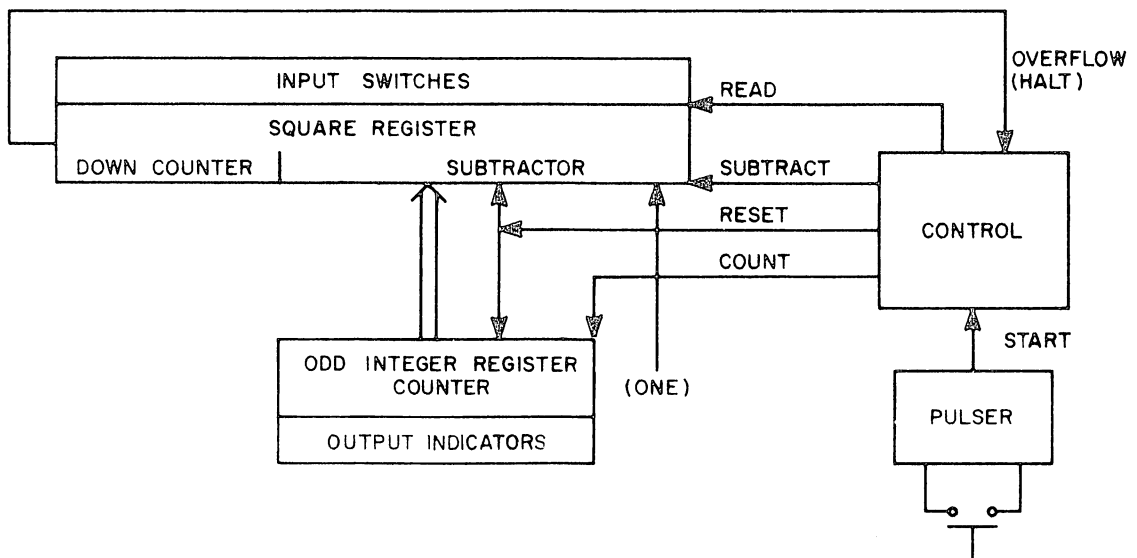


Figure 4 Final Simplified Block Diagram

The subtractor can be implemented with a two-step process similar to that used in the two-step adder of Experiment VIII. The first step is a half-subtract, the second step is a borrow which will ripple down through the more significant bits.

Figure 5 shows the logic diagram for the arithmetic-memory element. The up-counter and down-counter are quite familiar. Notice the similarity between subtractor logic and the adder logic of Figure 2 in Experiment VIII.

1. The borrow equation in the subtractor is:

$$\text{Borrow out} = (B + (S \oplus C)) (C + (S \oplus C))$$

where

B = Borrow in

S = Square register bit (minuend)

C = Counter bit (subtrahend)

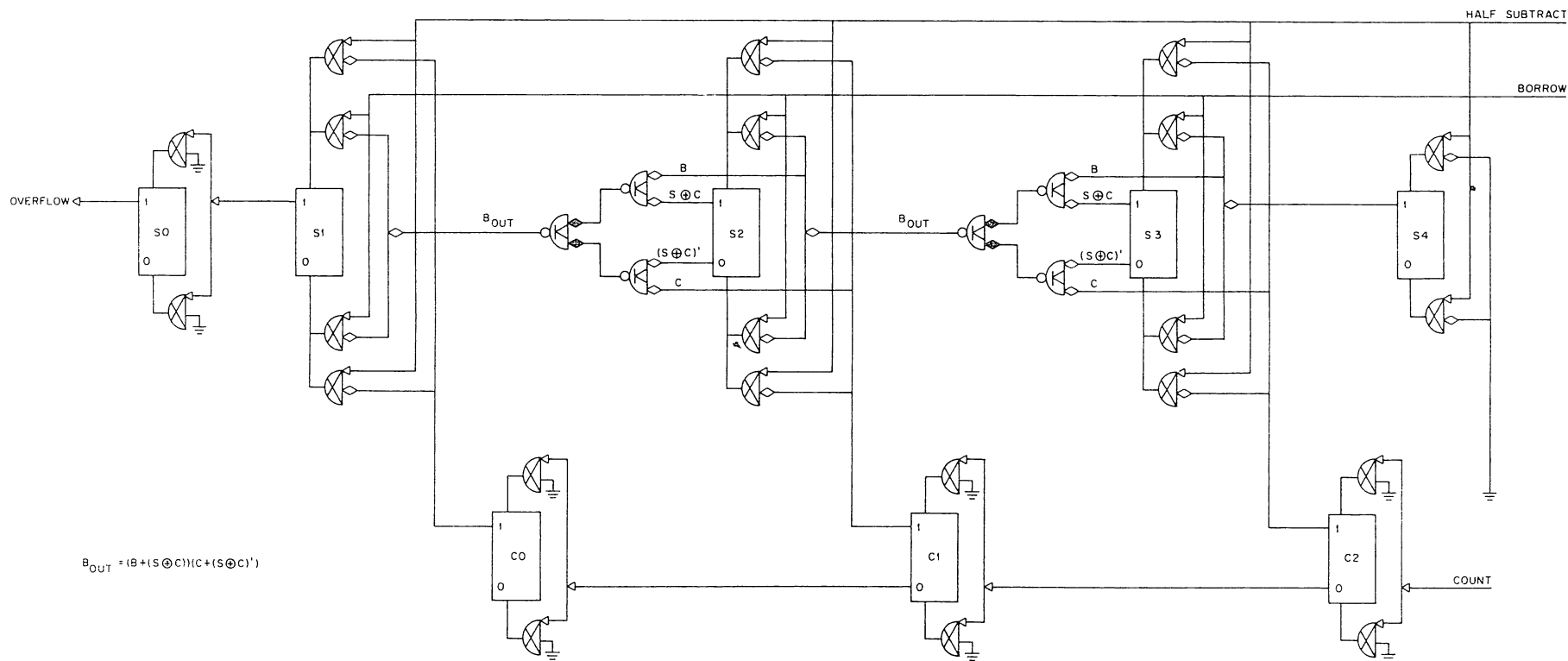


Figure 5 Arithmetic-Memory Element

Draw a subtractor truth table and, from this, derive an equation for the borrow out. Prove the equation above is the same as the one you derived.

Figure 6 shows the arithmetic and memory logic with the I-O added. The lights on the subtractor register and the read-in gates on the odd integer counter are for test purposes only. The lights on the odd integer counter show the output.

The read-in to the subtractor presents something of a problem. Normally, the read-in is performed by clearing and then conditionally reading ONEs from the toggle switches. In this case, however, reading a ONE would generate a borrow in the down counter circuitry.

In the up counter of Experiment XII, a similar problem occurred. Clearing a flip-flop to ZERO generated a carry. This was overcome by using a long clear pulse, then conditionally reading the ONEs, which did not generate carries.

To avoid errors due to unwanted borrows, the down counter must use the opposite procedure. All flip-flops are set to the ONE state with a long pulse that overrides the borrows. Then, the read-in conditionally clears the flip-flops to ZERO. Thus, a closed switch corresponds to a binary ZERO.

Figure 7 shows the control element. When the pushbutton is depressed, a ground signal is generated which resets the counter and square register. When the reset signal goes negative, it reads ZEROs into the subtractor and enters a pulse into the delay loop. This pulse generates the half-subtract and borrow, then checks for overflow. If the overflow has not arrived, it sends a pulse to the odd integer counter and again circulates around the loop. There is a pulse amplifier on the half-subtract line and an extra diode gate on the read line because of the large load that these pulses must drive.

2. Calculate the load which must be driven by:

- (a) The pulser
- (b) The gate which drives the read square line
- (c) The gate which drives the read counter line
- (d) The output of D1
- (e) The pulse amplifier
- (f) The output of D2
- (g) The output of D4

3. The setting for each delay unit must allow all the necessary action to be completed before the next pulse occurs. For example, D3 must be set for a long enough time that the borrow signal can propagate through all the stages of the down counter, set the overflow flip-flop, and disable the DCD gate on D4. This requires three flip-flop transition times plus one DCD gate setup time for a total of 640 nanoseconds.

Maximum Delay Through Any Circuit

Diode Gate	50 nanoseconds
Flip-Flop and DCD Gate	80 nanoseconds
Pulse Amplifier and DCD Gate	50 nanoseconds
DCD Gate Set-up Time	400 nanoseconds

Minimum Delay Through All Circuits Is Zero

Table 1 Circuit Delay Table

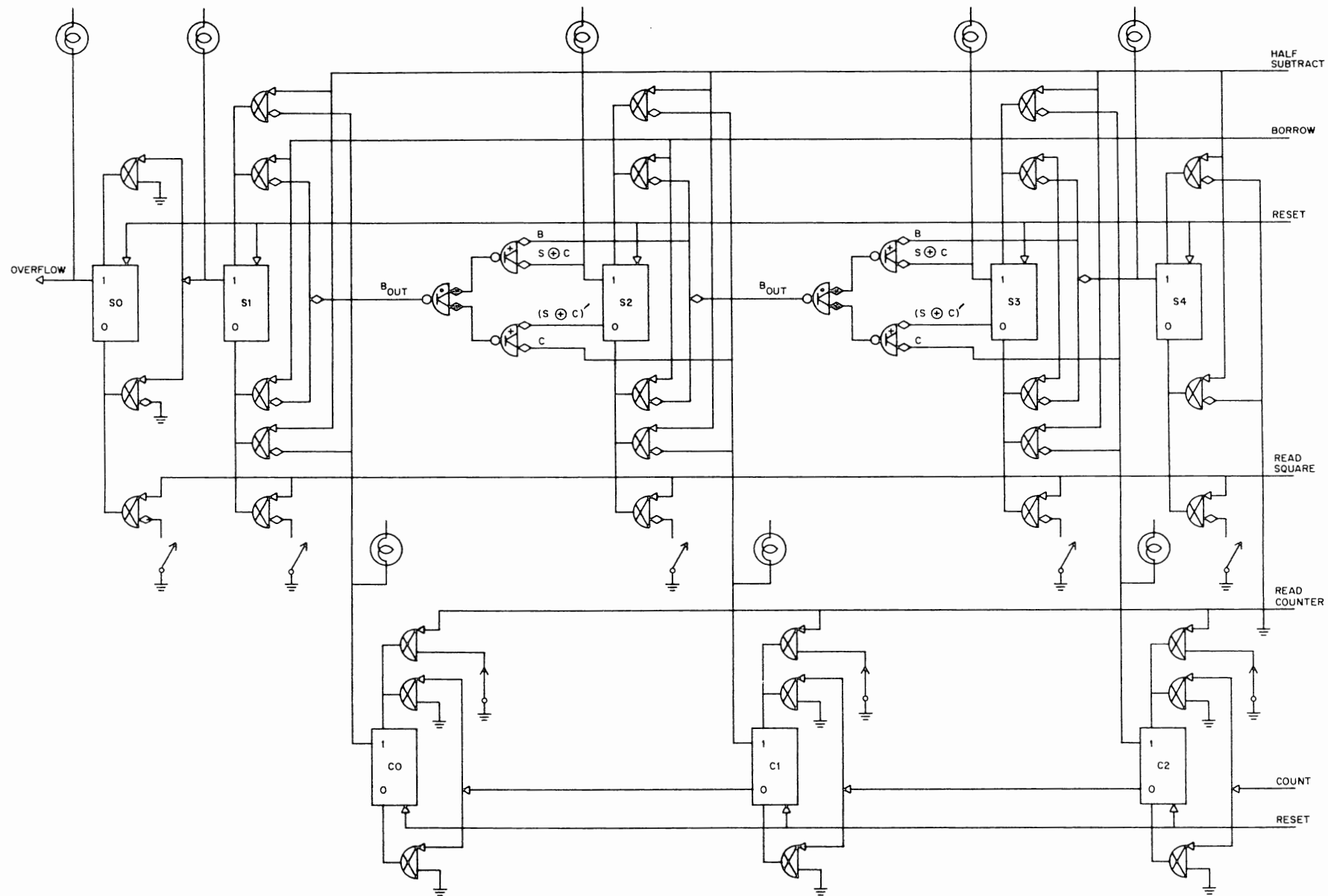


Figure 6 Arithmetic-Memory Element with IO Logic

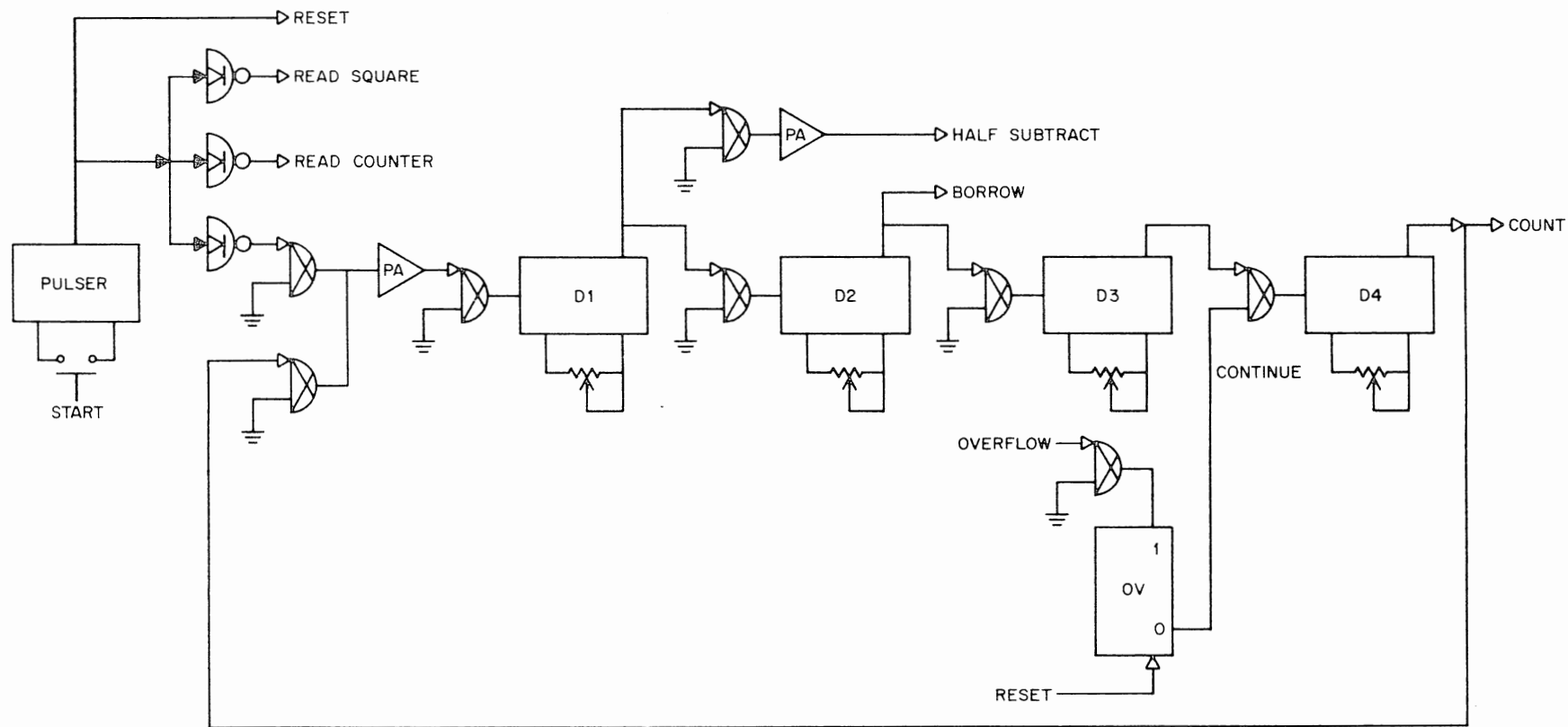


Figure 7 Control

Using the information from Table 1, calculate the minimum settings for:

- (a) D1
- (b) D2
- (c) D4

4. Construct and test the square root computer. Since there are over 100 wires involved, connections should be made carefully and the system should be checked at each step of the way. If such a procedure is not followed, you may discover that the complete system does not work and wonder which of the 100 wires is connected wrong.

- (a) Wire the odd integer counter. Use the temporary manual control logic shown in Figure 8 to test the counter for correct operation.

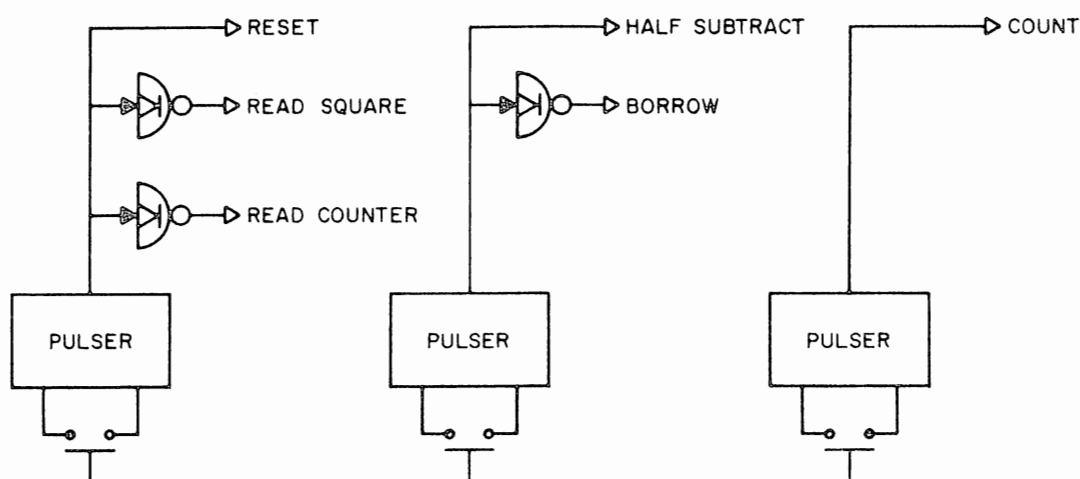


Figure 8 Temporary Manual Control

- (b) Connect the five square register flip-flops to the indicators. Then wire the reset and read lines. Using the temporary manual control signals, as illustrated in Figure 8, check that the square flip-flops read in correctly.
- (c) Complete all of the square register logic, and wire in the remaining signals from the temporary manual control logic. Reset. This should set the square register to 11111; the odd integer counter to 000, and the effective odd integer to 0001. Test that the borrow logic operates correctly by repeatedly subtracting 1. Then read test numbers into the odd integer register and test that the complete subtract logic operates correctly.
- (d) Remove the manual control circuitry and wire the automatic control circuitry. Do not connect the overflow line. Start the delay loop into operation with the pushbutton and test with the oscilloscope that it is operating correctly. Set each delay for one microsecond.
- (e) Connect the overflow line and your computer is complete. Test its operation. (If you find a difficulty that was not previously encountered, disconnect the delay units, then wire them in one at a time, starting with D1. In this way you can check each step of the operation.)

5. What is the square root of the following numbers?

- (a) 10000
- (b) 11010
- (c) 01101
- (d) 11111

PART 5 SPECIAL PROBLEMS

6. Combine your kit with one of your neighbor's. Using the circuits available in two laboratories, design and construct an expanded square root computer which will handle input numbers up to 10 bits. Be sure to calculate the loads presented to each of the control pulses and include pulse amplifiers wherever they are necessary. Calculate the minimum allowable time for the delay settings as described in question 3 and set your delay units for these minimum values.
7. Remembering that forming a square is the inverse of forming a square root and that subtraction is the inverse of addition, modify the square root computer logic so that it will calculate squares. Change only the I-O connections and the control circuitry.

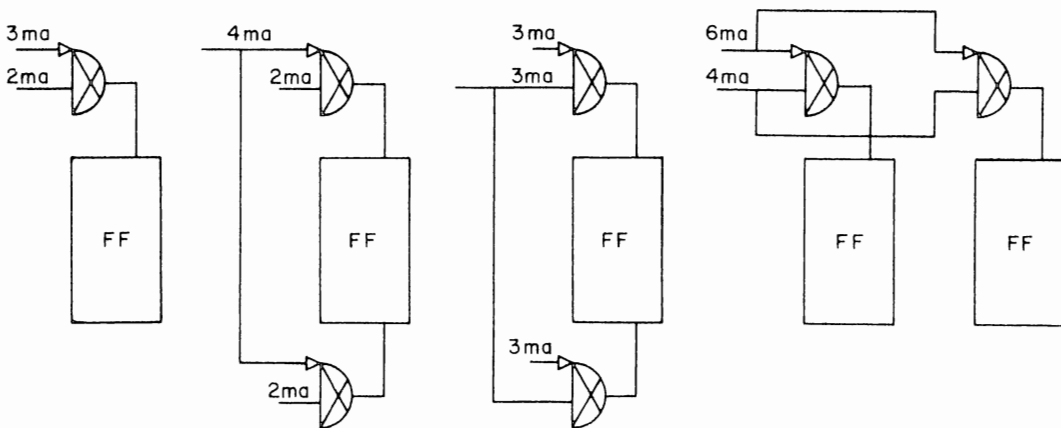
APPENDIX A

LOADING RULES

When interconnecting basic circuits to perform logical operations, it is important to keep the load on each circuit within its driving ability. The R series loading rules are simple because all inputs draw current from the same direction and because most circuits are built around diode and DCD gates.

The diode gate input draws 1 ma (milliampere) and the output drives 20 ma less 2 ma for the load resistor. Indicator inputs and pulser outputs are the same as diode gates.

The DCD gates draw 2 ma at the level inputs and 3 ma at the pulse inputs. When two DCD gates are driving both sides of the same flip-flop, the load on both pulse inputs totals only 4 ma. When the level inputs are tied together as in a complement configuration, the total input load is only 3 ma, as shown in the figures below.



A flip-flop is two diode gates cross-connected. The direct set and clear terminals draw 1 ma. The output will drive 20 ma less 2 ma for the load resistor permanently connected in the flip-flop and less 1 ma for driving the opposite side of the flip-flop, less the load of internal DCD gate connections.

The outputs of the delay one-shot, pulser and comparator are also similar to the diode gate. Each output can drive 20 ma less 2 ma for the load resistor. The outputs of the clock and pulse amplifier are more powerful. Each output can drive 73 ma less an internal load of 3 ma. The tables below summarize the loading.

Inputs	Load
Diode (including all direct inputs)	1
DAC	1
DCD Level	2
DCD Pulse	3
Special Load (above pulser)	10

Outputs	Load	Driving Ability
Diode Gate	2	18
Pulser	2	18
Delay One-Shot	2	18
Comparator	2	18
Flip-Flop (upper)	7	13
Flip-Flop (lower)	9	11
Clock	3	70
Pulse Amplifier	3	70

APPENDIX B

ADDING WAVEFORMS

To observe what is happening at one circuit point relative to another, the two waveforms may be added together by connecting each point to the oscilloscope probe through a 10 kohm resistor (see Figure 1).

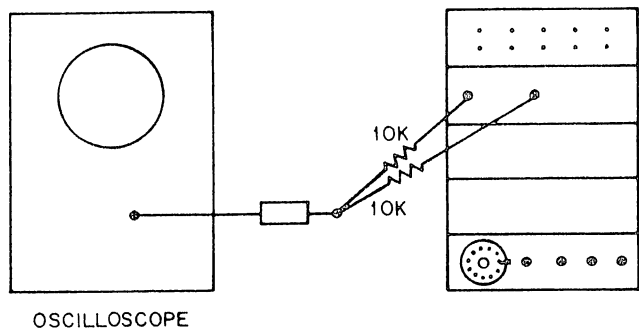


Figure 1

The individual waveforms should be observed separately, as the combined waveform may otherwise be difficult to interpret. Figure 2 shows some examples. In Figure 2(a) two flip-flops are operating at the same frequency but flip-flop B is slightly delayed with respect to A. Figure 2(b) shows the same situation except that the flip-flop outputs are out of phase. Figure 2(c) shows two flip-flops in a counter. Flip-flop A is the lesser significant bit. Since flip-flop A drives flip-flop B, there is a slight delay between the switching of the two circuits, resulting in small patterns similar to those seen in 2(a) and 2(b).

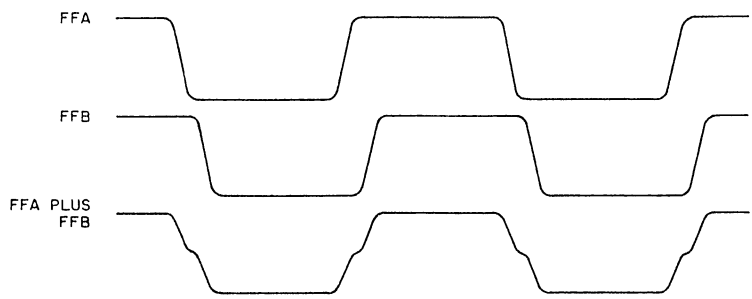


Figure 2(a)

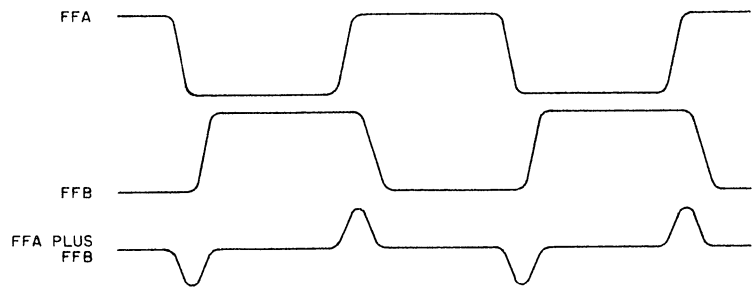


Figure 2(b)

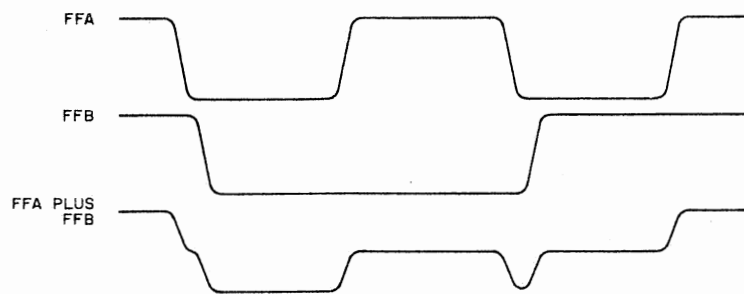


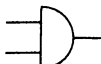


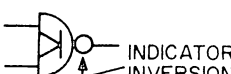

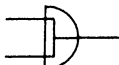
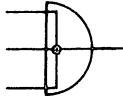
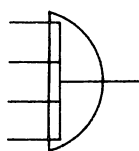
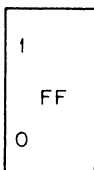
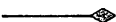
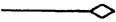
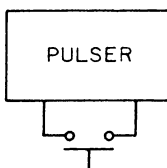


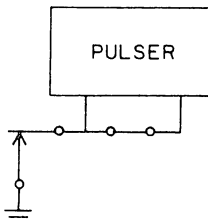


Figure 2(c)

APPENDIX C

SYMBOLS

Gate	Gate with Inputs	Gate with Inputs and Outputs	Gate with Inputs and Outputs When Output Has Inversion
			
Diode-Capacitor-Diode Gate (DCD)	NAND for -3v Assertion NOR for Ground Assertion		NOR for -3v Assertion NAND for Ground Assertion
			
Wired AND for -3v Assertion Wired OR for Ground Assertion			
			
(If ground is defined as the "true" or asserted level, this symbol means "wired OR." If -3v is the asserted level, the symbol means "wired AND.")			
Level Transition or Pulse, Ground-Going for Assertion	 Flip-Flop		
Level, -3v for Assertion  Level, Ground for Assertion 	 Pulser with Pushbutton		
 Clamped Load Resistor  Ground	 Pulser with Dial		

APPENDIX D

FLIP CHIP MODULES IN THE LOGIC
LABORATORY

The Logic Laboratory may be expanded with Digital's full line of FLIP CHIP modules as described in the Digital FLIP CHIP Module Catalog. Because the workbook is a teaching aid while the module catalog is a designer's tool, there are some differences in the descriptions which should be noted.

The flip-flop, as used in the workbook, has all terminals ground for assertion. In the catalog, flip-flop outputs are negative for assertion. Thus, there are pin connection differences as shown in Figure 1.

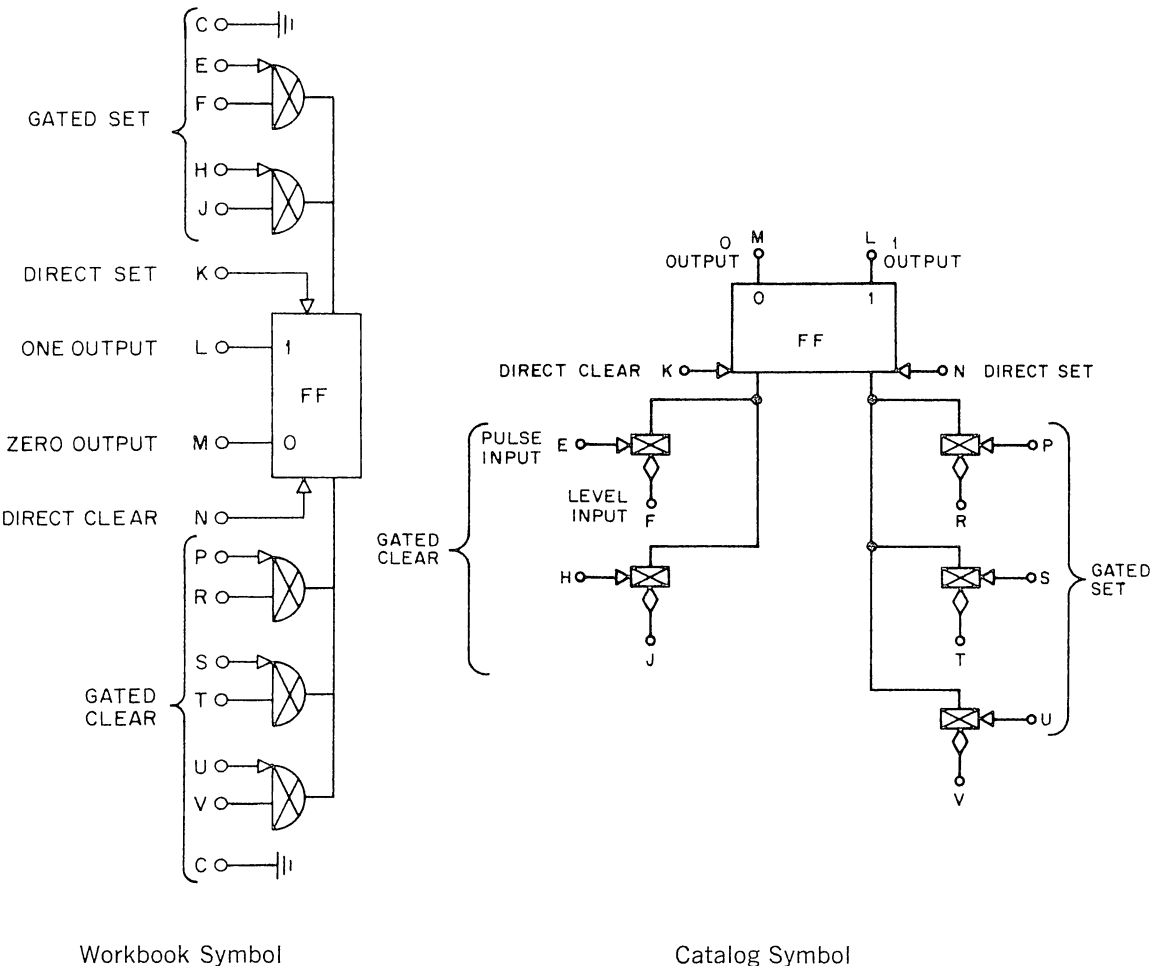
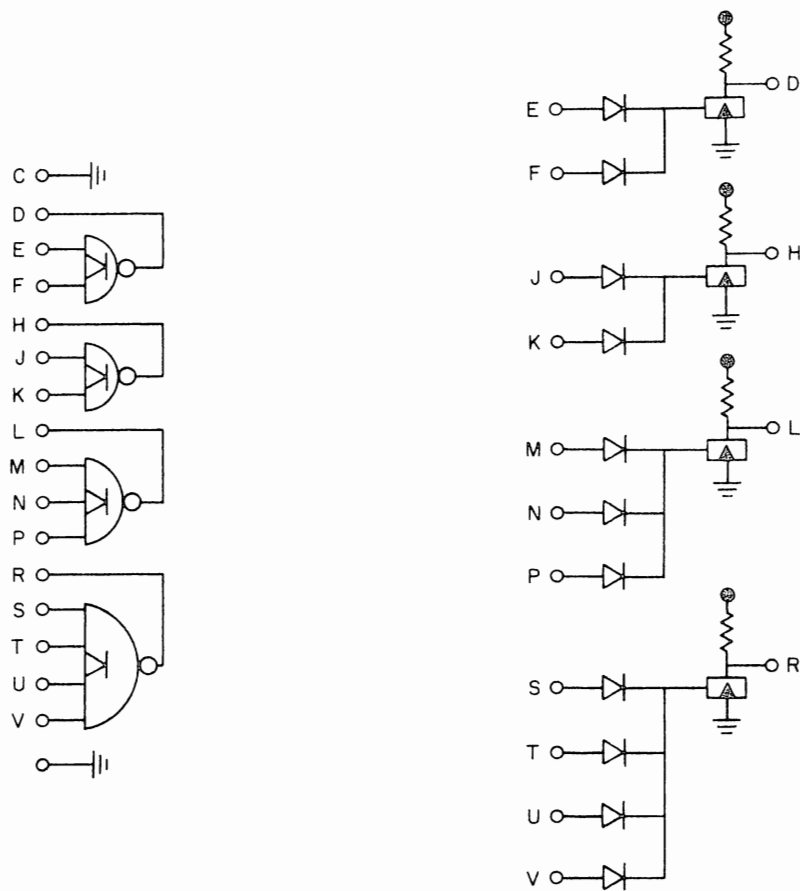


Figure 1 R201 Flip-Flop

The gate circuits, R121 and R122, used in the workbook have internal connections to 2-ma clamped load resistors.

Figure 2 shows the symbol for the R121 in workbook notation and catalog notation. The R122 is designed for teaching applications and does not have a catalog equivalent.

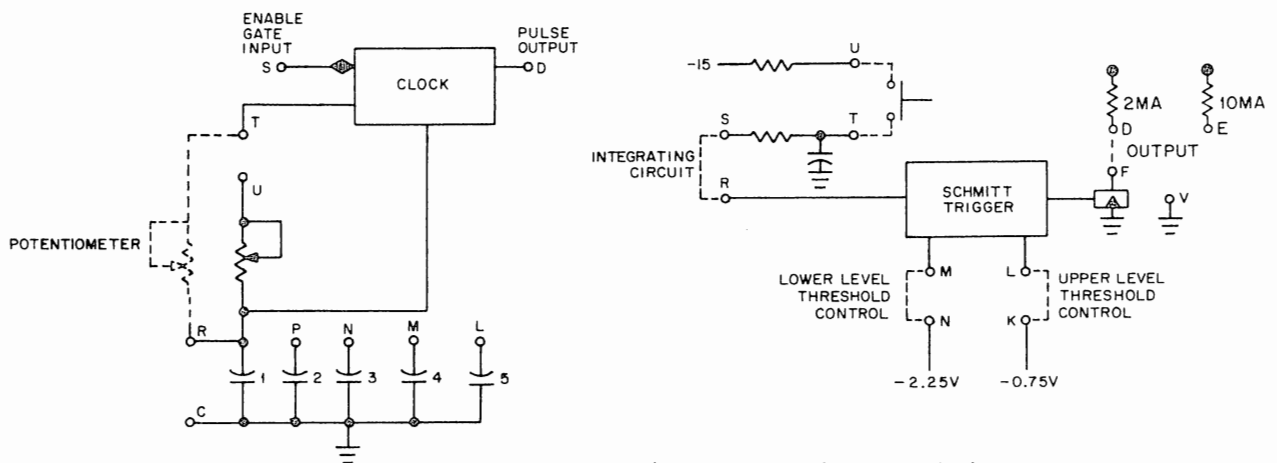


Workbook Symbol

Catalog Symbol

Figure 2 R121 Diode Gate

The pulser and clock circuits in the 700D Power Supply Input Panel utilize FLIP CHIP modules Type W501 and R401 respectively. Connections, as shown in Figure 3, are made on the panel. These modules may be removed and used separately as described in the module catalog.



700D Panel Connections Shown with Dotted Lines

Figure 3 700D Panel Connections

For additional information:

	Pages
Logic Laboratory Workbook	
General	1-12
	19- 44
FLIP CHIP Modules Catalog	
General	1-18,
	186-190
R Series	1-18
W Series	64-66
B Series	84-91,
	99-106
A Series	126-127
Hardware	136-140

APPENDIX E

NOTES TO THE INSTRUCTOR

WORKBOOK OUTLINE

Because the aim of this workbook is to complement a lecture course in digital logic, the material is presented here in as nearly as possible the same format and sequence as in the leading textbooks. However, since these textbooks vary considerably in their approach, particularly during the later sections, the instructor will probably wish to reorder the experiments to match more closely his chosen textbook.

As outlined in the preface, the workbook contains a set of seven core experiments. These consist of Experiments I, III, IV, V, VIII, XI, and XV together with the reading material from Parts 1 of VI and VII. Altogether, these cover number systems, circuits, Boolean algebra, addition, subtraction, control and practical considerations, the last experiment being a summary. For a shorter course, a good breaking point is just before the last two core experiments (XI and XV) which go into the more practical matters.

The remaining experiments are fairly flexible. Experiments XIII and XIV covering analog-digital conversion should be performed in that order. Experiments IX and X require a knowledge of BCD, either from the text or from Experiment II. The others are independent. With these restrictions, the eight non-core experiments can easily be moved around each other and to any later position among the core experiments.

The following notes and the Textbook-Workbook Cross Reference Table may be of assistance to the instructor in matching the experiments to his present course outline:

EXPERIMENTS I and II can accompany the introductory sections of the textbook where the details of circuitry have not yet been discussed. The student wires the circuits by rote and studies the properties of number systems. He also learns the advantages of the binary system and that a single circuit can be used for many functions. Parts 1 through 4 of Experiment I introduce the logic kit and the notation that will be used later on. Part 5 of Experiment I and all of Experiment II may be omitted or moved to a later section in the course if circuitry has been introduced in time to advance directly to Experiment III.

EXPERIMENTS III through VI introduce the circuits and the logic conventions that will be used later on. In Experiment III (gates) questions 4 through 7 may be omitted to shorten the experiment. Part 5 may be omitted if the student is not to do any design work. Part 6 (grounding) should definitely be included.

In Experiment IV, any of questions 5 through 11 may be omitted.

Experiment V introduces the dual polarity logic notation and the concept of wired gates, both of which are used throughout the later parts of the book. The questions at the end of this experiment give the student his first chance to truly design circuits to match Boolean expressions.

This is continued on into Part 1 of Experiment VI which applies the dual polarity logic convention to toggle switches and flip-flop outputs. The remaining parts of Experiment VI go into more details of Boolean equations and explain the notations that will be used later on. In the later chapters Boolean equations are given, but not

WORKBOOK-TEXTBOOK CROSS REFERENCE TABLE

		Text Books				
Workbook Experiments		Bartee, Thomas C. Digital Computer Fundamentals, McGraw-Hill Book Company, Inc., 1960, pages —	Chu, Yaohan, Digital Computer Design Fundamentals, McGraw-Hill Book Company, Inc., 1960, pages —	Ledley, Robert S. Digital Computer and Control Engineering, McGraw-Hill Book Company, Inc., 1960, pages —	Phister, Montgomery Jr. Logical Design of Digital Computers, John Wiley & Sons, Inc., pages —	Smith, Charles V. L., Electronic Digital Computers, McGraw-Hill Book Company, Inc., 1959, pages —
Numbers	I	29-47	1-15	1-69	16-29	1-29
	II	46-51	53-57	90-92	242-246	1-29
Circuits	III	54-59 61-63 65-69 73-81	160-163 168-179 194-196 200-202	645-676	21-25	85-101
	IV	81-104	194-196 185-194 368-378	676-679	25-27	107-125
	V	113-138	89-122 182-183	295-315 320-360	30-60	148-178
	VI		122-132 378-383		112-142	179-217
Adders	VII	147-162 169-172	363-371	485-506	242-257 276-278	218-226
	VIII	162-169 172-180	18-24 383-391	519-528	258-266 278-285	226-233
	IX	180-187	53-59	506-510	266-276	
	X	257-287	78-87	242-253 747-756	326-339 399-401	390-406
Code Conversion						
Control	XI	99-106	209-213 392-393	543-573		363-389
Timing	XII				212-221	407-410
Analog-Digital Conversion	XIII	286-287		739-756	229-230 238	
	XIV			739-756	229-230 238	
Computer	XV					
SUMMARY						

stressed, so the instructor may ask questions along these lines, or not, as he wishes. If he does not wish to stress Boolean equations, it would be sufficient for the student to simply read Part 1 of Experiment VI.

EXPERIMENTS VII, VIII and IX cover addition and subtraction. Part 1 of Experiment VII introduces the addition equations. Following the approach of most textbooks, serial addition and subtraction are introduced next. However, if time is limited, it is suggested that students skip directly from Part 1 of Experiment VII into Experiment VIII which describes parallel adders and subtractors. The two-step parallel adder is the basic circuit described here, and it is used extensively in subsequent chapters. The later parts of Experiment VIII go into positive and negative numbers and may be covered or not as time allows.

Experiment IX covers BCD addition and subtraction following the techniques learned from the two-step parallel binary adder. For this experiment, the student should have some background in BCD codes either from Experiment II or from the text. The final two questions are quite elaborate, and could be done either in place of the experiment or as an end-of-term, special project.

EXPERIMENT X (code conversion) stands somewhat by itself. It requires a knowledge of the dual polarity logic convention and Part 2 requires a knowledge of BCD codes. This experiment could be used either to study the implementation of Boolean expressions or to accompany a lecture series on input-output. The last problem is extensive enough to be done in place of the experiment or as a special project at the end of the term.

EXPERIMENT XI (control circuitry) introduces two new circuits, the pulse amplifier and the delay one-shot. Thus, it provides background material for the remaining four experiments as well as a description of control circuitry. Question 11, which requires a knowledge of the two-step parallel adder, would be suitable for an end-of-term project. The rest of the experiment requires only knowledge of the dual polarity logic convention and so may be moved to any position after Experiment VI.

EXPERIMENT XII is devoted primarily to practical considerations — the problems of simultaneous signals and synchronization of random signals. It must be preceded by Experiment VI Part 1 and Experiment XI. Although some references turn up in subsequent sections, this can be omitted with a few comments from the instructor. The last problem is suitable for a special project.

Experiments XII, XIII and XIV go into more detail than most of the textbooks, hence they might be used during the lecture series on large scale memory elements or any other topic that is not treated in the workbook.

EXPERIMENTS XIII and XIV cover the techniques of digital-to-analog and analog-to-digital conversion. These require Experiment XI as background. They are not referred to in the last chapter so they may be omitted. If the section is to be shortened, it is recommended that Parts 1, 2 and 3 of Experiment XIII and Part 1 of Experiment XIV be retained.

EXPERIMENT XV is a summary of the material learned in the core experiments (binary numbers, circuits, gating, 2-step parallel addition, and control). In this experiment the student combines his knowledge to actually construct a special-purpose computer. It is, of course, a small computer so the introductory sections of this chapter explain how this computer fits in with the overall picture of computers today. This is designed as an end-of-term project with the questions preceding the actual experiment, so that the student can, if desired, hand in his final report at the time he comes in to do the experiment. For the student who wishes to tackle independent design work, this could be replaced by any of the special project questions from the earlier experiments, or by either of the last two problems in this experiment.

THE LOGIC LABORATORY IN A PROGRAMMING OR SURVEY COURSE

The Logic Laboratory may also be used in courses where only a few laboratory periods are devoted to computer hardware. In this case the student may construct counters and adders to learn the binary number system and gain a feeling for the operation of computer circuits. The following sections are recommended:

Session I	Experiment I
Session II	Experiment II
Session III	Read Experiment VII, Part 1 and perform Experiment VIII, Parts 1-4 with help from the instructor on question 8.

NOTATION

The logic symbols used here are designed for simplicity in drawing and emphasis on the practical considerations involved in using real physical circuits. Thus, the logical (or one bit time) delay necessary for the correct operation of parallel logic is indicated on the symbols with an X. The gate functions and flip-flops states are undefined on the panel symbols so that the student will learn their various functions by writing in the definitions from the workbook.

For simplicity the symbols use a single polarity logic system, with ground as the assertion level and ground-going pulses. (Users of Digital's Flip-Chip Modules will notice that this differs from the standard Digital dual polarity convention, as detailed in the Appendix D.) Where inversion (or inhibit) is required by the circuitry, the symbol adopted by the American Standards Association is employed — a small circle. Beginning with Chapter V, the student learns to define his own polarity convention with solid and hollow diamonds, thereby applying De Morgan's law at each step of the logic.

EXPERIMENTAL EQUIPMENT

THE LOGIC LABORATORY. The experiments in this workbook use the Logic Laboratory manufactured by Digital Equipment Corporation, Maynard, Massachusetts. There are three general equipment configurations and innumerable variations in between these. The quantity, model number and type of equipment included in the general configurations is:

STANDARD LOGIC LABORATORY

(Suitable for Experiments I-X except for some optional questions.* Provides 30 hours of laboratory study. Covers the basic principles of digital logic.)

1	H901	Module Mounting Panel
1	H902	Indicator — Switch Panel
1	W052	Indicator Driver
6	R201	Flip-Flops
3	R121	Quadruple NOR Gates
1	R122	Quadruple NAND Gate
4	911-2"	Box of 2" Patch Cords
5	911-4"	Box of 4" Patch Cords
2	911-8"	Box of 8" Patch Cords
1	911-16"	Box of 16" Patch Cords
1	700D	Power Supply and Signal Generator Panel (with Modules)
1	4913	Mounting Rack

ADVANCED LOGIC LABORATORY

(Suitable for Experiments I-XII and XV. Provides 39 or more hours of laboratory study. Covers the basic principles of digital logic and practical considerations.)

1		Standard Logic Laboratory
1	H901	Module Mounting Panel
3	R201	Flip-Flops
1	R121	Quadruple NOR Gate

* Specifically, these are Exp. VIII, Q5; Exp. IX, Q11 and Q12; and Exp. X, Q4, all of which require an extra R121 gate module. Also the serial adder of Experiment VII should be reduced to a two-bit version with the standard Logic Laboratory.

2	R302	Dual Delay One-Shots
1	R602	Dual Pulse Amplifier
1	911-4"	Box of 4" Patch Cords
1	911-8"	Box of 8" Patch Cords
1	911-16"	Box of 16" Patch Cords
1	911-32"	Box of 32" Patch Cords

ADVANCED LOGIC LABORATORY WITH A-D

(Suitable for all experiments. Provides 45 or more hours of laboratory study. Covers same material as Advanced Logic Laboratory plus analog-digital conversion techniques.)

1		Advanced Logic Laboratory
1	H903	Analog-Digital Conversion Panel (with modules)

OSCILLOSCOPE. The logical operation of the circuits is studied by using the push-button or dial as an input and observing the outputs on the indicator lights. This provides simultaneous monitoring of all the important points in the circuitry at the end of each logical step.

To study the electrical operation of the circuits, the clock is used to provide high frequency input signals and the output is monitored on an oscilloscope. An oscilloscope with a single trace, AC coupled input and a bandwidth of 5 megacycles is suitable for observing the various timing relationships and actions that are performed by the circuits. However, a higher frequency oscilloscope is recommended if available, since it will give minimum distortion to the actual shape of the waveforms and allow more precise measurement of the delay times.

Since there are many component parts in a digital circuit, it is frequently necessary to display more than one waveform. Here a dual trace plug-in is useful, if available. If not, the waveforms may be observed simultaneously by simply adding them through two 10K resistors as described in Appendix B.

COMPONENTS. Four potentiometers are included on the indicator-switch panel to provide variable resistances. In a few questions the student is asked to connect an external capacitor or resistor to the circuit. In this case the component lead can be bent over in a V-shape and inserted in the stacking input of the miniature banana jack.

digital

**EQUIPMENT
CORPORATION**

MAYNARD, MASSACHUSETTS

Cambridge, Mass. • Washington, D. C. • Parsippany, N.J.
Rochester, N.Y. • Los Angeles • Palo Alto • Chicago • Ann
Arbor • Pittsburgh • Denver • Huntsville • Orlando
Carleton Place and Toronto, Ont. • Reading, England • Paris,
France • Munich and Cologne, Germany • Sydney, Australia